الدرس السابع:



-تحديد الـ Vertexes في الـTransformed
-رسم مثلث بي الـTransformedColored

تحديد الـ Vertexes في الـTransformed

قلنا في الدرس السابق بأنه عندما نتكلم عن الـ Transformed فمعناه ضمنياً أنني سأستخدم نقاط البكسل (Pixels) لتحديد إحداثيات الجسم المراد رسمه على شاشة الكمبيوتر, وأوضحنا أن نقطة البداية (Origin Point) تكون في أقصى الزاوية اليسار, كما في الشكل بالأسفل:

0,0	pixel	pixel	pixel	pixel
pixel	pixel	pixel	pixel	pixel
pixel	pixel	pixel	pixel	Pixel
pixel	pixel	pixel	pixel	Pixel
pixel	pixel	pixel	pixel	Pixel

X
Y

الآن لنقل أنني أريد أن أرسم مثلث ونعلم أن المثلث له ثلاثة نقاط إحداثياتها كالتالي :

النقطة الأولي0 = x = 0 , y :
النقطة الثانية200 = x = 0 , y :
النقطة الثالثة200 = x = 200 , y :

0,0	pixel	0,200	pixel	pixel
pixel	pixel	pixel	pixel	pixel
pixel	pixel	200,200	pixel	Pixel
pixel	pixel	pixel	pixel	Pixel
pixel	pixel	pixel	pixel	pixel

X
Y

ما رأيك الآن أن نحول الكلام الذي بالأعلا برمجياً إلى الـ:DirectX
الخطوة الأول :التصريح عن الـ TransformedColored والذي هو أحد أنواع الـ Transformed وهو يعمل كالوعاء الذي نضع إحداثيات النقاط والألوان المراد الرسم بها, ولجعل الأمور أسهل سنجعلة مصفوفة (Array)لتسهل عملية ترتيب البيانات (إحداثيات النقاط), وسنعطية أي إسم وليكنvertexes

كود:

```
CustomVertex.TransformedColored[] vertexes ;
```

الخطوة الثانية: عمل داله (function) خاصة بي الـ vertex, وهو مهم فقط من أجل ترتيب الكود وجعل إدارته أفضل, سننشئ بداخلها كائن (Object) للـ vertexes الذي تكلمنا عنه بالأعلا, ونحدد بداخله عدد النقاط المراد رسمها, وهي كما في الشكل بالأعلا ثلاثة ليكون كالتالي:

كود:

```
vertexes = new CustomVertex.TransformedColored[3];
```

أي أن الرقم ثلاثة يمثل عدد النقاط المراد رسمها, وكما نعلم بأن الترقيم في المصفوفات يبدأ من الرقم صفر.

النقطة الأول [0:]

كود:

```
vertexes[0].X = 0;
vertexes[0].Y = 0;
vertexes[0].Color = Color.White .ToArgb();
```

النقطة الثانية:[1]

كود:

```
vertexes[1].X = 200;
vertexes[1].Y = 0;
vertexes[1].Color = Color.Blue .ToArgb();
```

النقطة الثالثة[2:]

كود:

```
vertexes[2].X = 200;
vertexes[2].Y =200;
vertexes[2].Color = Color.Green .ToArgb();
```

نلاحظ بعد أن حددنا إحداثيات النقاط في المحوريين x و y قمنا بإضافة الجمل التالية:

كود:

```
vertexes[0].Color = Color.White .ToArgb();
vertexes[1].Color = Color.Blue .ToArgb();
vertexes[2].Color = Color.Green .ToArgb();
```

بواسطة هذه الجملة نستطيع إعطاء لون لأي نقطة, فكما تذكر بأننا حددنا الـ **vertex** من نوع **TransformedColored** والذي بدورة يدعم الالوان, حيث أن النقطة الأولي تأخذ اللون الأبيض, والثانية ستأخذ اللون الأزرق, والثالثة ستأخذ اللون الأخضر, أما بالنسبة للدالة **ToArgb()**فهي إختصار على لنوع اللألوان المستخدمة والتي تمثل.**(red, green, blue)**

ليصبح الكود لدي كالتالي:

كود:

```
public void tranvertex()
{
vertexes = new
CustomVertex.TransformedColored[3];

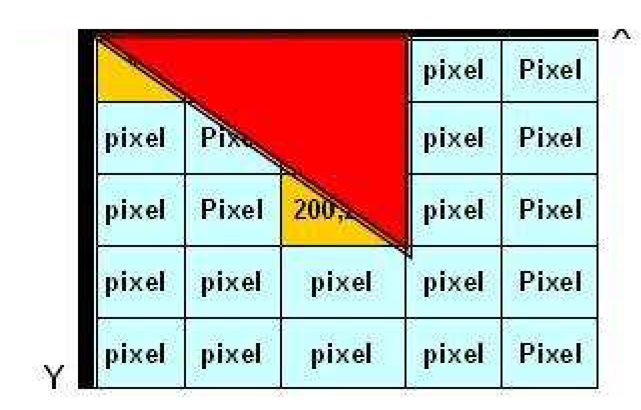

vertexes[0].X = 0;
vertexes[0].Y = 0;
vertexes[0].Color = Color.White .ToArgb();


vertexes[1].X = 200;
vertexes[1].Y = 0;
vertexes[1].Color = Color.Blue .ToArgb();


vertexes[2].X = 200;
vertexes[2].Y =200;
vertexes[2].Color = Color.Green .ToArgb();

}
```

الخطوة الثالثة: نقوم بتوصيل بين هذه النقاط لنحصل على مثلث كما في الشكل بالأسفل:



لتوصيل هذه الخطوط ببعضها البعض يعني أننا سنقوم بعملية الرسم, والدالة الخاصة بالرسم هي ()OnPaint والتي تكلمنا عنها في الدروس السابقة, أي أن أي شئ نريد رسمه سنصرح عنه في هذه الدالة.
هنا سنقوم بالتصريح بداخل الـ ()OnPaint عن الدالتين VertexFormatو الـ ,DrawUserPrimitives وسيكون الكود بشكل مشابه لهذا..

كود:

```
Public OnPaint()
{
.....
.....
VertexFormat();
DrawUserPrimitives();
.....
.....
}
```

الآن لنأتي إلى قصة هاتان الدالتان:
الأولى وهي الـ VertexFormat تكمن فائدتها بانها تخير الـ DirectX بنوع الـ Vertex المراد الرسم به وهو في مثالنا الـ TransformedColoredوسيكون برمجياً كالتالي:

كود:

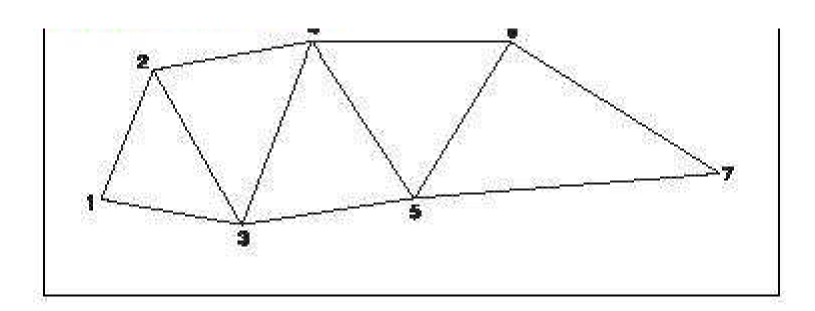
```
device3d.VertexFormat = CustomVertex.TransformedColored.Format ;
```

الثانية وهي الـ DrawUserPrimitives تكمن فائدتها بأنها تحدد طريقة الرسم حيث أنها تحوى ثلاثة براميترات: (Parameters)
الأول: من أجل تحديد كيفية رسم الخطوط
خاصية TriangleList أي توصيل كل نقطة بالآخرى (كما سنفعل في هذا المثال.)

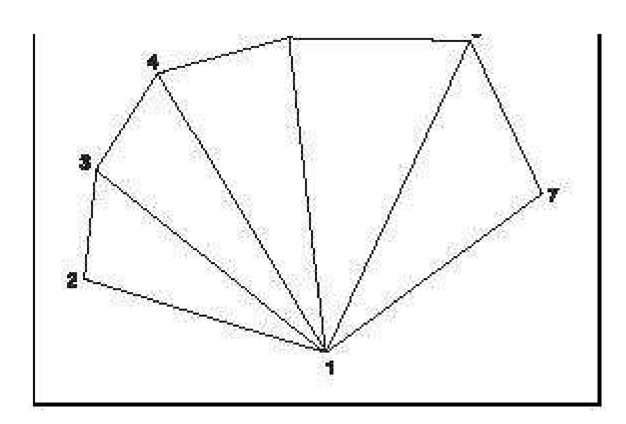
كود:

```
device3d.DrawUserPrimitives( PrimitiveType.TriangleList  ,.. ,..);
```

خاصية TriangleStrip سيكون التوصيل بشكل شبكة كما في الشكل بالأسفل



خاصية TriangleFan سيكون التوصيل بشكل مروحة حيث كل النقاط ستشترك بنقطة واحدة كما في الشكل بالأسفل:

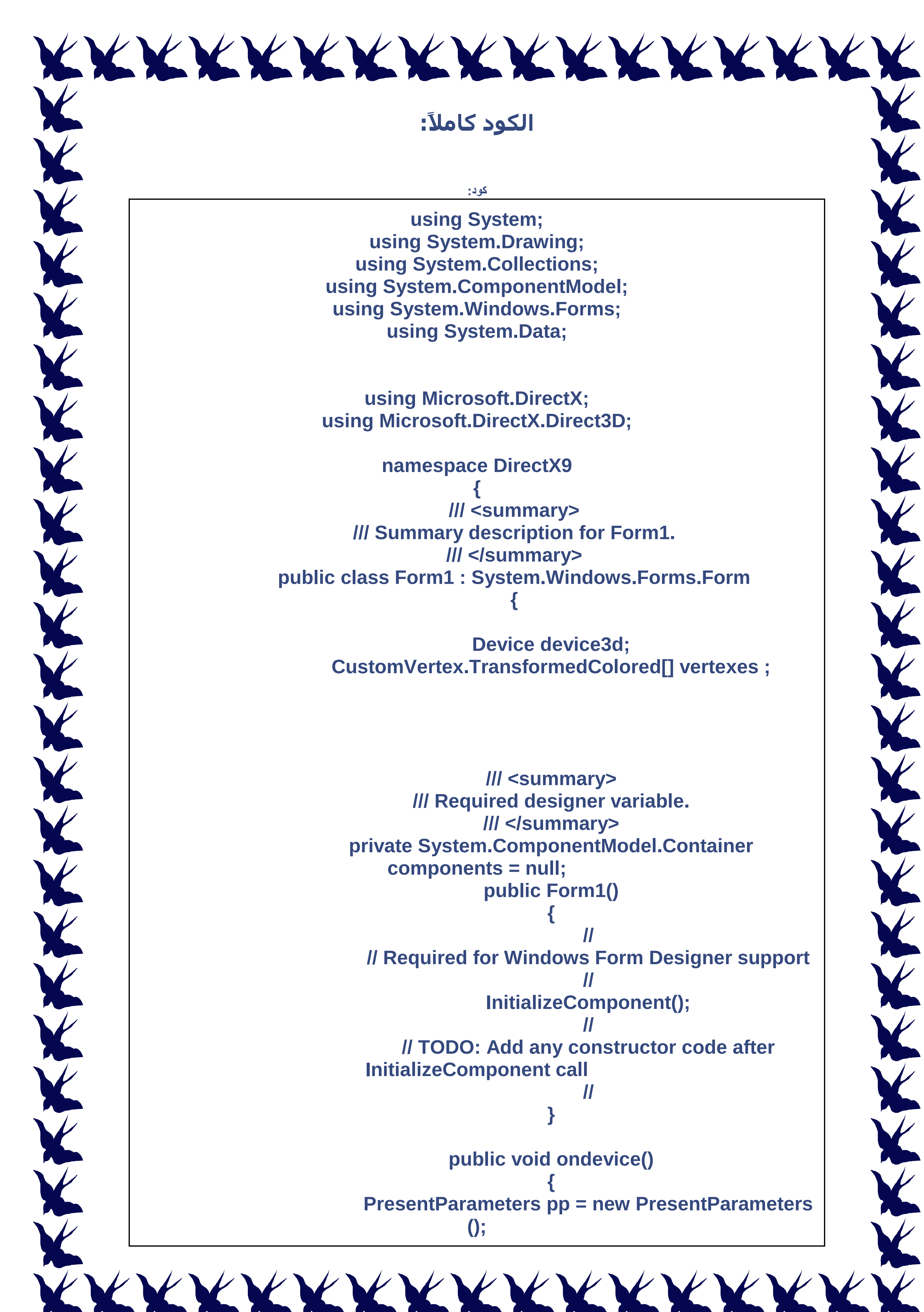


الثانية : لتحديد عدد المثلثات المراد رسمهم وهو في مثالنا 1
الثالثة : الكائن الذي قمنا بإنشائة في الأعلا وهوvertexes
ليصبح الكود بالشكل التالي:

كود:

```
device3d.DrawUserPrimitives( PrimitiveType.TriangleList  , 1, vertexes );
```

رسم مثلث بي الـTransformedColored



الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;


using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

namespace DirectX9
{
/// <summary>
/// Summary description for Form1.
/// </summary>
public class Form1 : System.Windows.Forms.Form
{

Device device3d;
CustomVertex.TransformedColored[] vertexes ;



/// <summary>
/// Required designer variable.
/// </summary>
private System.ComponentModel.Container
components = null;
public Form1()
{
//
// Required for Windows Form Designer support
//
InitializeComponent();
//
// TODO: Add any constructor code after
InitializeComponent call
//
}

public void ondevice()
{
PresentParameters pp = new PresentParameters
();
```

```
pp.SwapEffect = SwapEffect.Discard;

pp.Windowed = true;


device3d = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);
}


public void tranvertex()
{
vertexes = new
CustomVertex.TransformedColored[3];


vertexes[0].X = 0;
vertexes[0].Y = 0;
vertexes[0].Color = Color.White .ToArgb();


vertexes[1].X = 200;
vertexes[1].Y = 0;
vertexes[1].Color = Color.Blue .ToArgb();


vertexes[2].X = 200;
vertexes[2].Y =200;
vertexes[2].Color = Color.Green .ToArgb();

}


protected override void OnPaint
(System.Windows.Forms.PaintEventArgs e)
{
device3d.Clear (ClearFlags.Target ,Color.Red
,1,0);
device3d.BeginScene();

// draw the triangle
device3d.VertexFormat =
CustomVertex.TransformedColored.Format ;
device3d.DrawUserPrimitives(
PrimitiveType.TriangleList  , 1, vertexes );

device3d.EndScene();
device3d.Present ();
}
```

```
/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

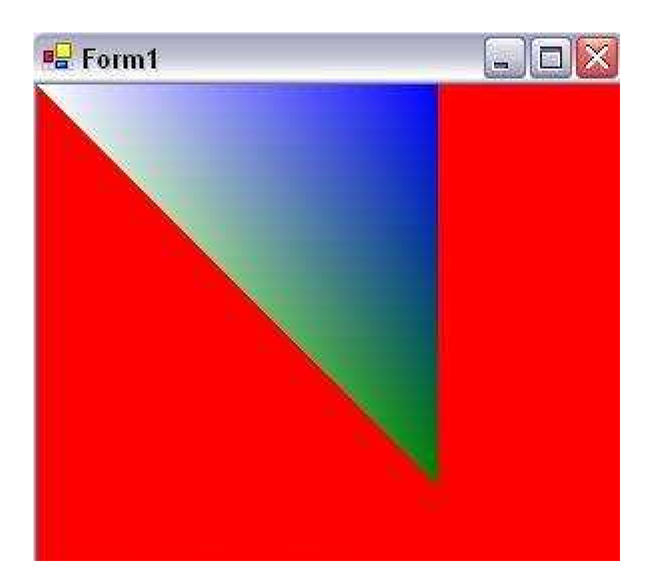
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
this.components = new
System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion

/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondevice ();
xx.tranvertex ();
Application.Run(xx);
}

}
}
}
```

عند عمل Compile للكود F5





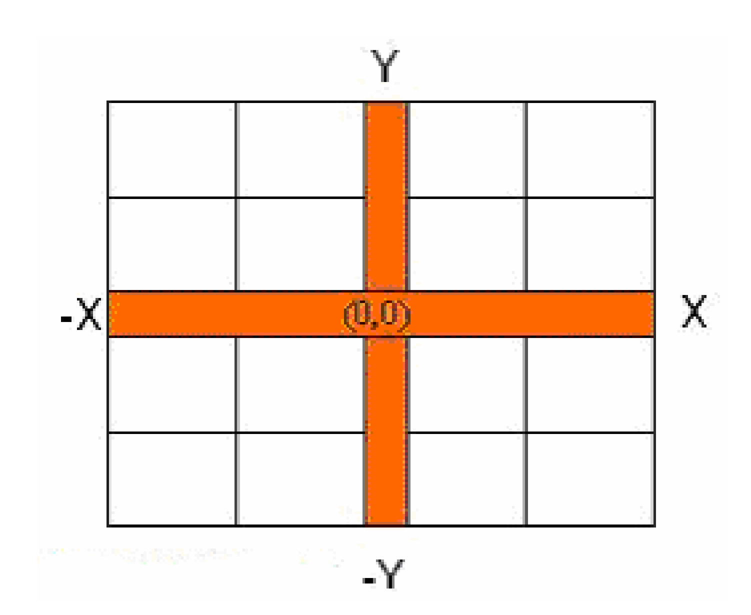
المبرمج هاني العزاوي
Hasa8384@yahoo.com



الدرس الثامن:
-تحديد الـ Vertexes في الـPositionColored
-الـCamera

تحديد الـ Vertexes في الـPositionColored

تكون نقطة الأصل (Origin Point) بي هذا النوع في منتصف الشاشة, وليس مثل الـ Transformed والتي تكون بأعلى يسار الشاشة (راجع الدرس السابع), وأيضاً من الممكن إعطاء اللوان لهذه النقاط أنظر إلى الشكل بالأسفل لتتوضح الفكرة أكثر..

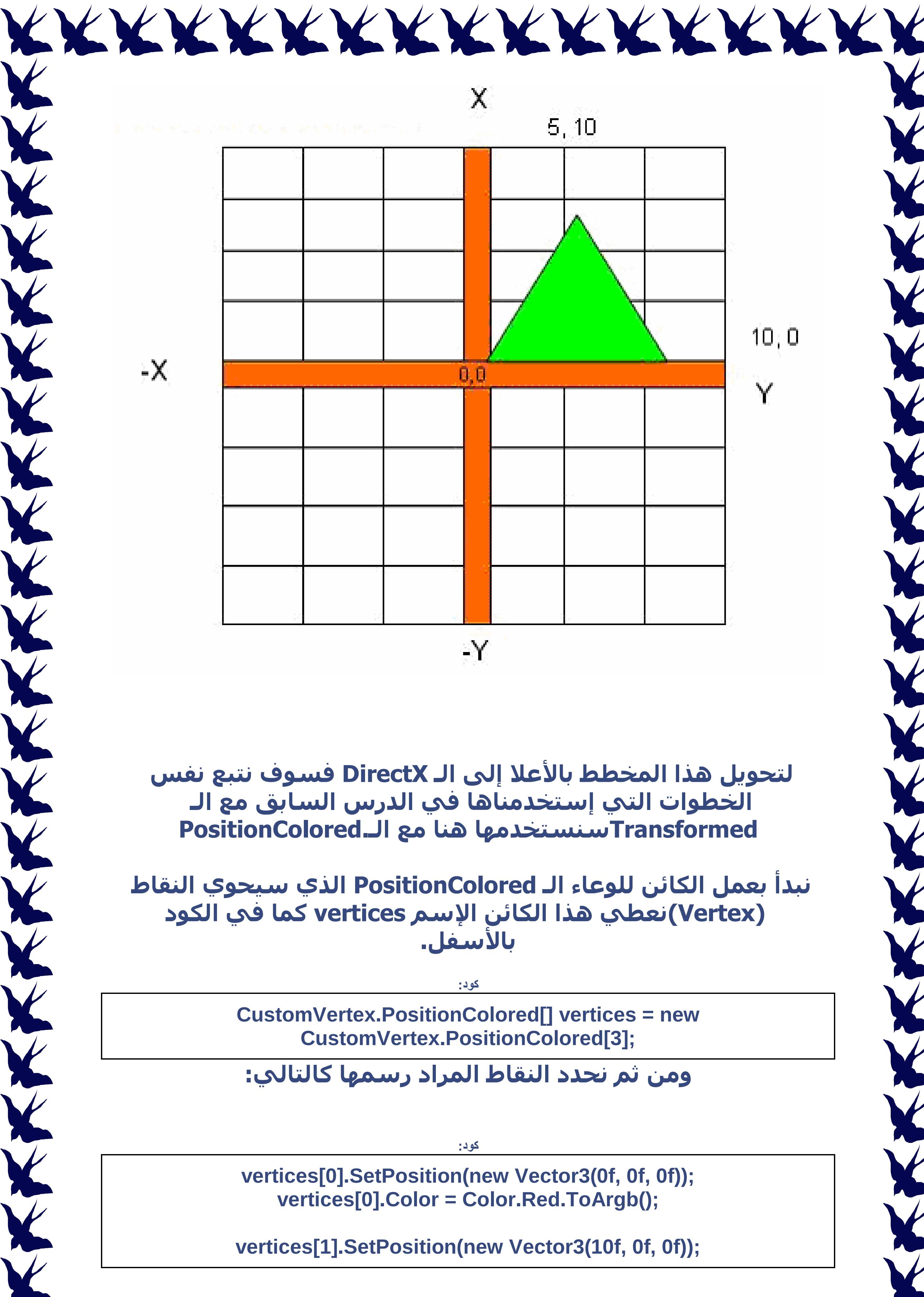


الآن لنرى كيف من الممكن رسم مثلث يحوي الإحداثيات التالية:

النقطة الأولي: x = ٠ , y = ٠ , z = ٠

النقطة الثانية: x = ١٠ , y = ٠, z = ٠

النقطة الثالثة: x = ٥ , y = ١٠ , z = ٠



لتحويل هذا المخطط بالأعلا إلى الـ DirectX فسوف نتبع نفس الخطوات التي إستخدمناها في الدرس السابق مع الـ Transformedسنستخدمها هنا مع الـPositionColored

نبدأ بعمل الكائن للوعاء الـ PositionColored الذي سيحوي النقاط (Vertex)نعطي هذا الكائن الإسم vertices كما في الكود بالأسفل.

كود:

```
CustomVertex.PositionColored[] vertices = new
CustomVertex.PositionColored[3];
```

ومن ثم نحدد النقاط المراد رسمها كالتالي:

كود:

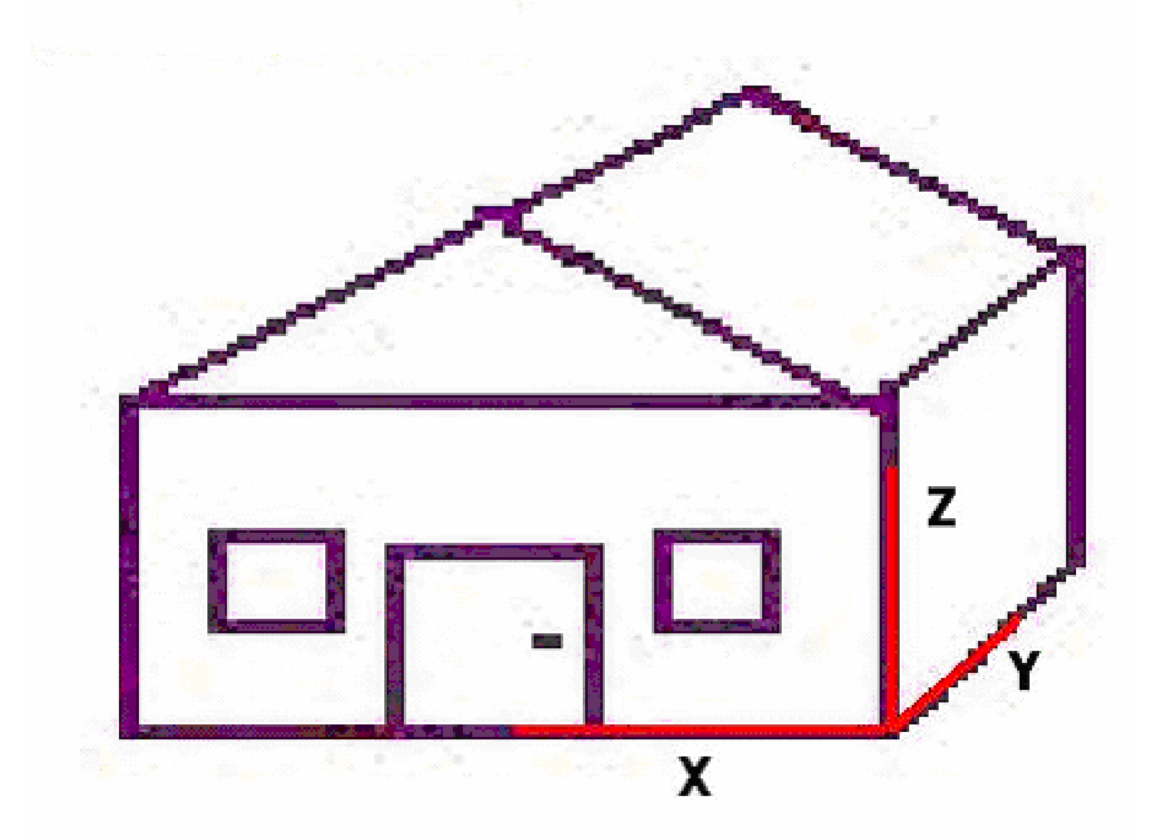
```
vertices[0].SetPosition(new Vector3(0f, 0f, 0f));
vertices[0].Color = Color.Red.ToArgb();

vertices[1].SetPosition(new Vector3(10f, 0f, 0f));
```

```
vertices[1].Color = Color.Green.ToArgb();

vertices[2].SetPosition(new Vector3(5f, 10f, 0f));
vertices[2].Color = Color.Yellow.ToArgb();
```

ما نراه جديد في هذا الكود هو وجود الفئة (Class) المسمى Vector٣والتي تعني تحديد ثلاثة إحداثيات وهو بالترتيب الأول على محور الـ x والثاني على محور y والثالث على محور الـ .z نستطيع القول بأن المحور z يمثل الإرتفاع عن السطح, أنظر إلى الشكل بالأسفل:



بقي لدينا التصريح عن الـ Vertex بداخل دالة الـ OnPaint وهي بنفس طريقة الـ Transformed التي تكلمنا عنها في الدرس السابق غير أنه هنا نقوم بإستبدالها بي الـ CustomVertex.PositionColored

كود:

```
device.VertexFormat = CustomVertex.PositionColored.Format;
device.DrawUserPrimitives(PrimitiveType.TriangleList, 1, vertices);
```

الكود كامل:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace DirectX_Tutorial

{

public class WinForm : System.Windows.Forms.Form

{

private Device device;
private System.ComponentModel.Container components = null;

public WinForm()

{
InitializeComponent();
this.SetStyle(ControlStyles.AllPaintingInWmPaint |
ControlStyles.Opaque, true);

}

public void InitializeDevice()

{

PresentParameters presentParams = new
PresentParameters();

presentParams.Windowed = true;

presentParams.SwapEffect = SwapEffect.Discard;

device = new Device(0, DeviceType.Hardware, this,
CreateFlags.SoftwareVertexProcessing, presentParams);

}
```

```
protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)

{

CustomVertex.PositionColored[] vertices = new
CustomVertex.PositionColored[3];

vertices[0].SetPosition(new Vector3(0f, 0f, 0f));
vertices[0].Color = Color.Red.ToArgb();

vertices[1].SetPosition(new Vector3(10f, 0f, 0f));
vertices[1].Color = Color.Green.ToArgb();

vertices[2].SetPosition(new Vector3(5f, 10f, 0f));
vertices[2].Color = Color.Yellow.ToArgb();

device.Clear(ClearFlags.Target, Color.DarkSlateBlue , 1.0f, 0);

device.BeginScene();

device.VertexFormat =
CustomVertex.PositionColored.Format;

device.DrawUserPrimitives(PrimitiveType.TriangleList, 1,
vertices);

device.EndScene();

device.Present();

this.Invalidate();

}

protected override void Dispose (bool disposing)

{

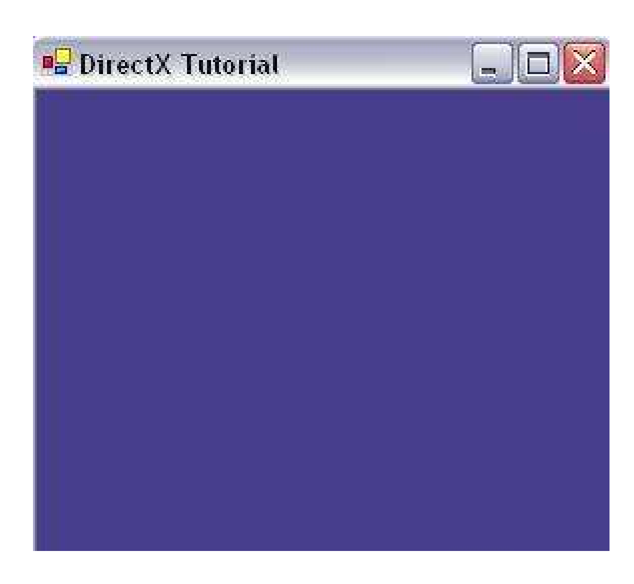
if (disposing)
```

```
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose(disposing);
}

private void InitializeComponent()
{
this.components = new System.ComponentModel.Container();
this.Size = new System.Drawing.Size(500,500);
this.Text = "DirectX Tutorial";
}


static void Main()
{
using (WinForm our_directx_form = new WinForm())
{
our_directx_form.InitializeDevice();
Application.Run(our_directx_form);
}
}
}
}
```

عند عمل run للكود في الأعلا, سيظهر لنا الشكل التالي:



ولأكن أين المثلث ؟؟؟؟

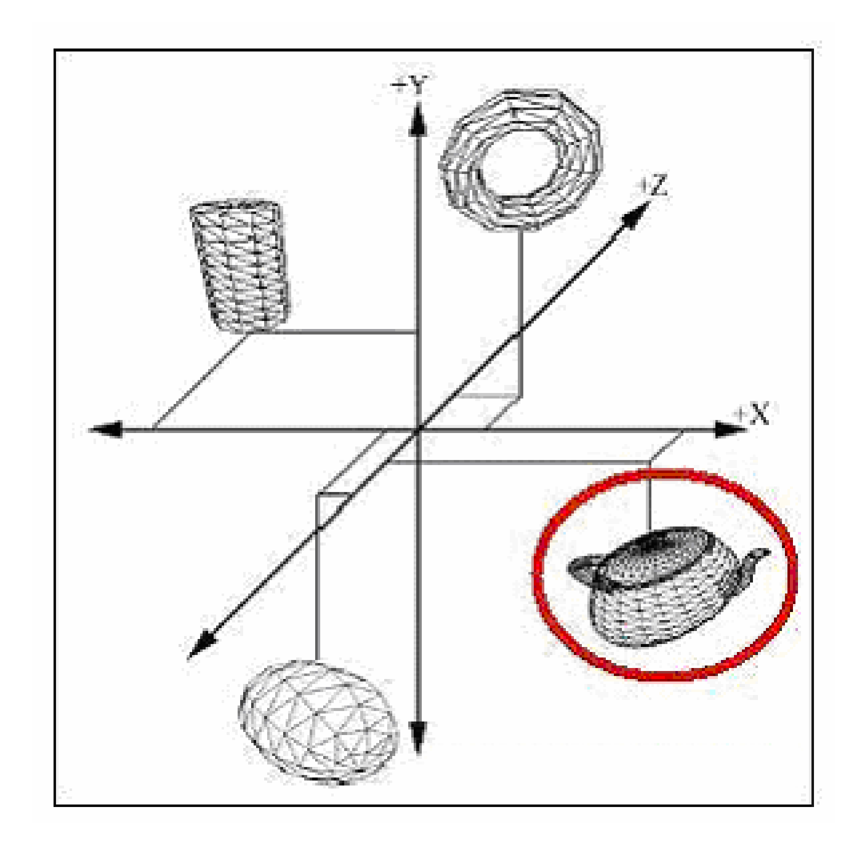
في الدرس الماضي عندما تكلمنا على الـ Transformed قمنا بتحديد الإحداثيات عن طرق البكسل (Pixels) أما عند إستخدام الـ PositionColoredفتصبح جميع الإحداثيات في الـ World Space أي مثل ما أنك في الفضاء الواسع, لذلك يجب تحديد النقاط المراد النظر إليها بإستخدام الكميرا...

....

-الـCamera

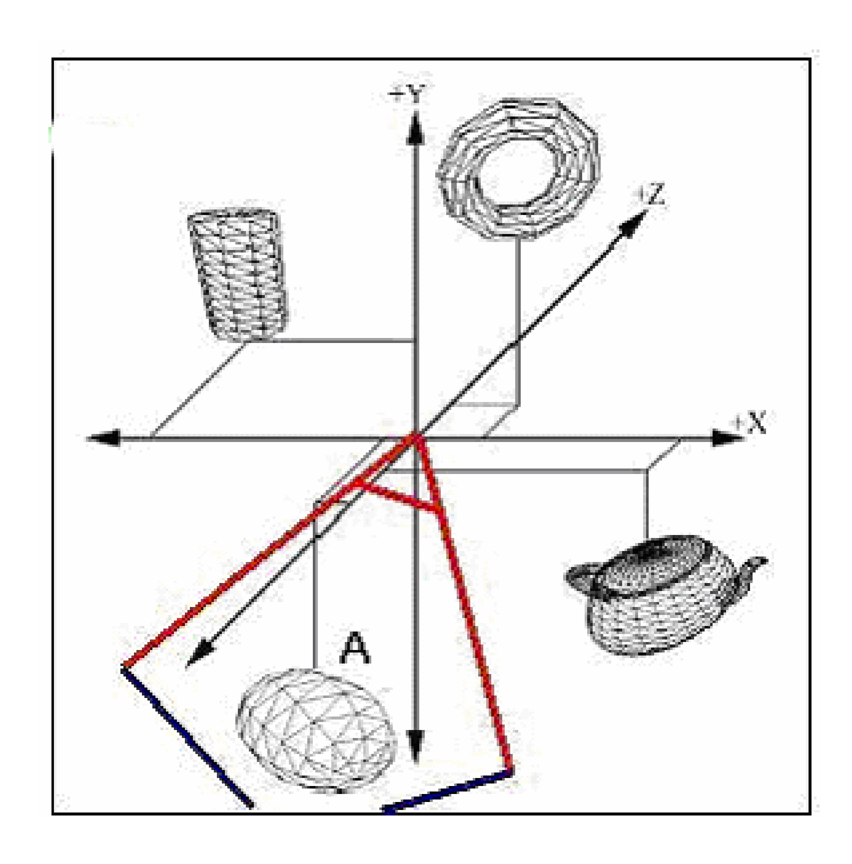
وهي من أهم المواضيع في برمجة الـ DirectX, عندما نريد التعامل مع الكميرا فيجب علينا أن نضع في ذهننا عدة أشياء منها:
أولاً: تنقسم عملية رسم الأشكال إلى مرحلتين الأولي هي:
:(Local Space)ويتم فيه رسم الشكل بي إحداثيات النقاط (Vectors)على المحور x و y و z الخاص بي الشكل نفسه, إليك الشكل بالأسفل لتوضيح ما أعنية:

: (World Space)وهي عملية نقل الجسم إلى الفضاء الحقيقي, حيث يأخذ بالحسبان الموقع والحجم والإتجاه بالنسبة لباقي الأجسام, أنظر إلى الشكل بالأسفل:

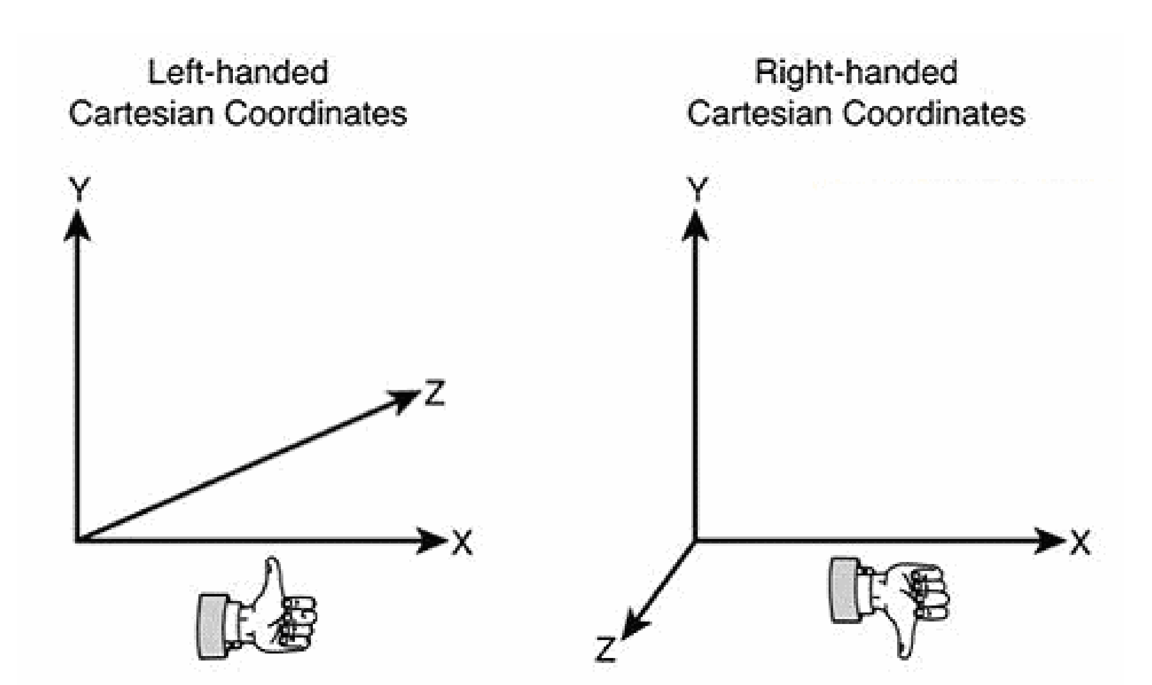


ثانياً :تكمن وظيفة الكميرا بي تحديد مجال الرؤيا في الـ World)

أنظر إلى الشكل بالأسفل, حيث قمنا بوضع الكميرا (Space) ,
المتمثلة بالشكل المخروطي (الإسقاط) من أجل إظهار الشكل A
فقط.



ثالثاً :عندما نريد تحديد موقع الشكل المخروطي (الكميرا) يجب علينا
أن نأخذ بالحسبان الـ:
PerspectiveFovRHوالتي تشير إلى الجهه اليمنى من محور
الإحداثيات أنظر إلى الشكل بالأسفل حيث يكون الإبهام إلى الأسفل
موازي للمحور.Z
PerspectiveFovLHوالتي تشير إلى الجهه اليسرى من محور
الإحداثيات أنظر إلى الشكل بالأسفل حيث يكون الإبهام إلى الأعلا
موازي للمحور.Z



الشكل الشائع والمستخدم هو الـ PerspectiveFovLH لأنه يمثل التوضع الحقيقي للأحداثيات.

ولتوضيحها أكثر.. دعنا نأخذ الأحداثيات في المثال بالأعلا وهي...

كود:

```
vertices[0].SetPosition(new Vector3(0f, 0f, 0f));
vertices[1].SetPosition(new Vector3(10f, 0f, 0f));
vertices[2].SetPosition(new Vector3(5f, 10f, 0f));
```

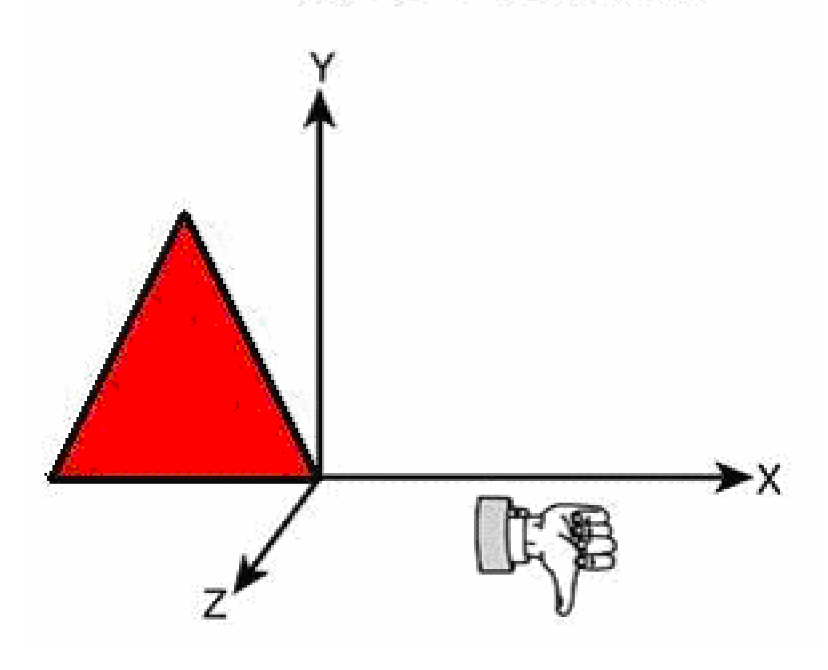
إذا قمنا بتمثيل هذه الإحداثيات وإستخدمنا الـ PerspectiveFovLH فستكون جهة رؤية الإحداثيات بوضعها الطبيعي كالتالي:



كود:

```
device.Transform.Projection = Matrix.PerspectiveFovLH();
```

أما إذا إستخدمنا الـ PerspectiveFovRH فستكون جهة الرؤية بالعكس كالتالي:



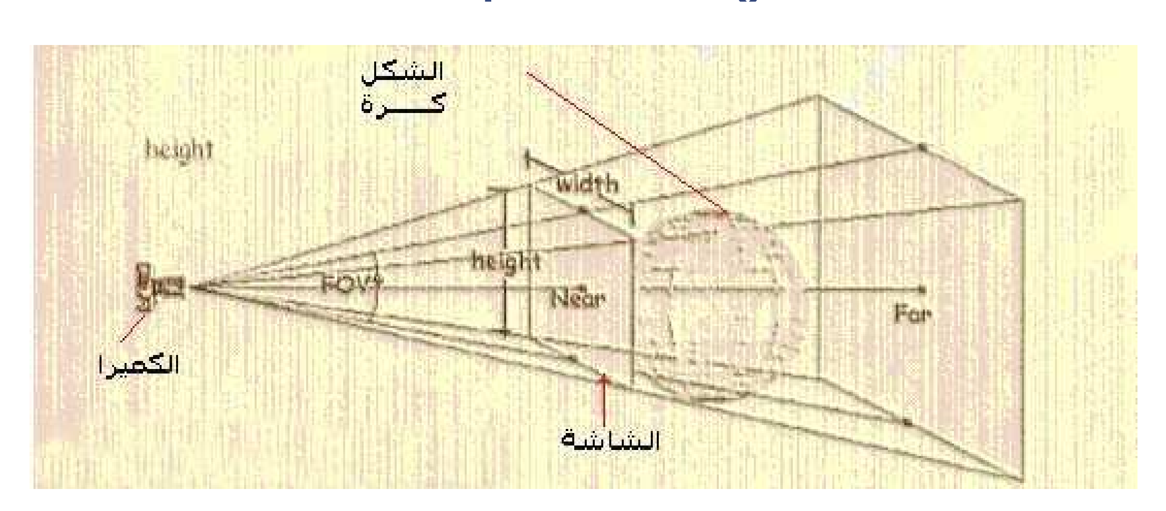
كود:

```
device.Transform.Projection = Matrix.PerspectiveFovRH ();
```

رابعاً :قمنا في الخطوة السابقة بتحديد جهة الرؤية, بقي الآن أن نحدد مدى الرؤية لدى الكميرا أي عملية الإسقاط (Projection), توجد عدة طرق من أجل تنفيذ عملية الإسقاط ,ولأكن أكثرها فاعلية هو الإسقاط المنظوري (Perspective Projection) لأنه يقوم بإسقاط الأشكال الهندسية بحيث تبدو الأشكال البعيدة عن الكميرا أصغر من تلك القريبة من الكميرا, رياضياً تستخدم مصفوفة معقدة لذلك, ولأكن وفر علينا الـ DirectX هذا العناء وذلك بإستخدام الداله PerspectiveFovRH()أوPerspectiveFovLH()

منطقة
توضع
الأشكال
Near
Fare
الزاوية
الأفقية
مركز الإسقاط
الزاوية
العامودية

ألا يذكرك الشكل بالأعلا بشئ... حاول أن تتذكر ... ألم تفتح جهاز التلفاز من قبل .. أعتقد بأنك بدأت تفهمني .... نعم فإذا فتحنا جهاز التلفاز فسوف نري بأن الشاشة موصولة بي قمع (شكل مخروطي) يشبة الشكل بالأعلا ... من هنا أخذ الـ DirectX بإستخدام نفس نظرية المنظور الذي يستخدمة التلفاز وإستطاع محاكاتها رياضياً بإستخدام إحدى هاتين الدالتين ()PerspectiveFovLH أو ()PerspectiveFovRH



تأخذ هاتان الدالتان البارامترات (Parameters) التاليه :
-زاوية الرؤية : من الشكل بالأعلا هي (FOV) وهي الخاصة بأمور الـ Zooming, وتستخدم في الألعاب الحربية حيث يقوم اللاعب بتحويل

نظام الرؤيه إلى العدسة المكبرة الموجودة على سلاحة ليستطيع إقتناص الأعداء. وغالباً ما تكون موضوعة بي زاوية PI/٤ حيث أن الـ PIيساوي (١٨٠)أي تصبح ١٨٠/٤ وتساوي ٤٥ درجة وهي نفس الدرجة التي يستخدمها جهاز التلفاز, حيث نلاحظ بأنه يوجد خاصية في بعض التلفزيونات الحديثة إمكانية عمل ZOOM لمشهد معين, الآن أصبحنا نعلم كيف يقوم بذلك, نعم... عن طريق تصغير الزاوية, فكلما صغرت الزاوية كلما كبر الجسم وذلك لأن صغر الزاوية يضغط على الجسم ليقترب إلى الأمام مما يؤدي إلى كبر حجمة تستخدم عمليات رياضية معقدة من أجل ذلك وهي ليست في مجال بحثنا.

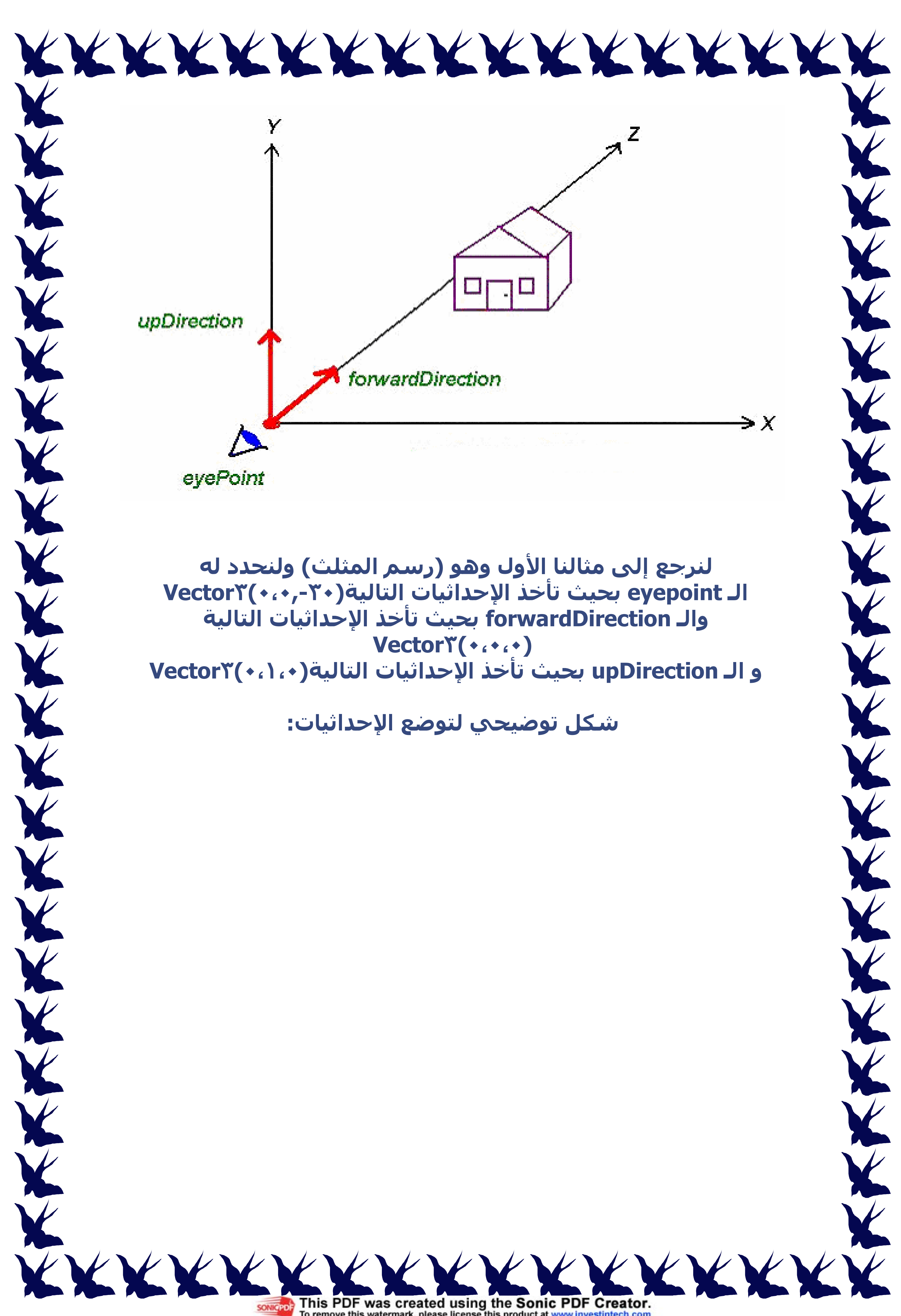
-الـ : Near وهي شاشة التي نعمل عليها .. أي الفورم, والذي بدورة يتكون من الطول والعرض ,وغالباً ما نقوم بتقسم العرض على الطول , width/height فهذا يساعد على تأقلم الشكل المرسوم بحسب الفورم عند تكبير وتصغير الشاشة.
-الـ : Far وهو عمق الجسم والذي يساعد على تحديد الجسم القريب والبعيد عن الكميرا.

لتصبح المعادلة كاملة كالتالي:

كود:

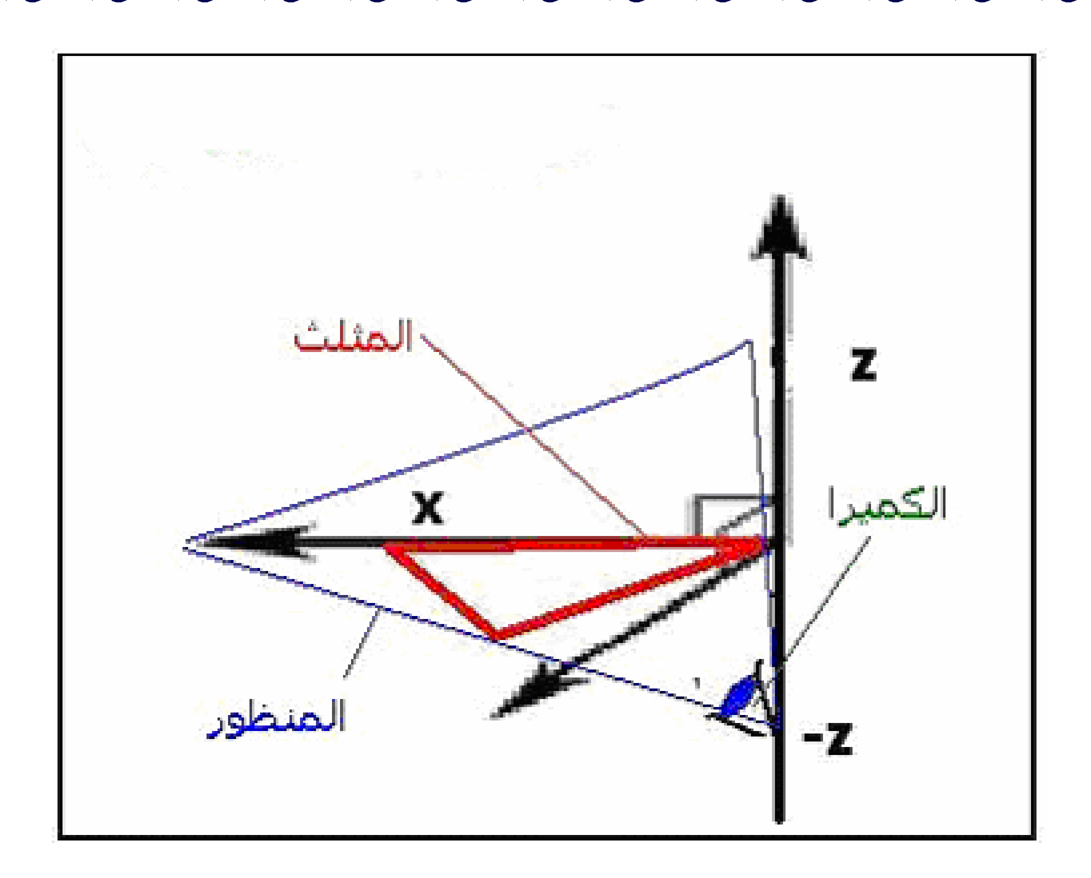
```
device.Transform.Projection = Matrix.PerspectiveFovLH((float)Math.PI/4,
this.Width/this.Height, -1f, 1f);
```

خامساً :بعد أن حددنا مكان الكميرا, والمنظور لها, بقي أن نحدد خواصها وذلك بإستخدام الدالة LookAtLH والتي تحوي ثلاثة بارامترات.(Parameters)
-الأول: يحدد موقع الكميرا (eyePoint) في الـWorld Space
-الثاني: الجسم المراد النظر إلية (الهدف(forwardDirection) ,(
-الثالث: تحريك الكميرا إلى أعلا.(upDirection)
أنظر إلى الشكل بالأسفل:



لنرجع إلى مثالنا الأول وهو (رسم المثلث) ولنحدد له
الـ eyepoint بحيث تأخذ الإحداثيات التالية Vector٣(٠،٠,-٣٠)
والـ forwardDirection بحيث تأخذ الإحداثيات التالية
Vector٣(٠،٠،٠)
و الـ upDirection بحيث تأخذ الإحداثيات التالية Vector٣(٠،١،٠)

شكل توضيحي لتوضع الإحداثيات:

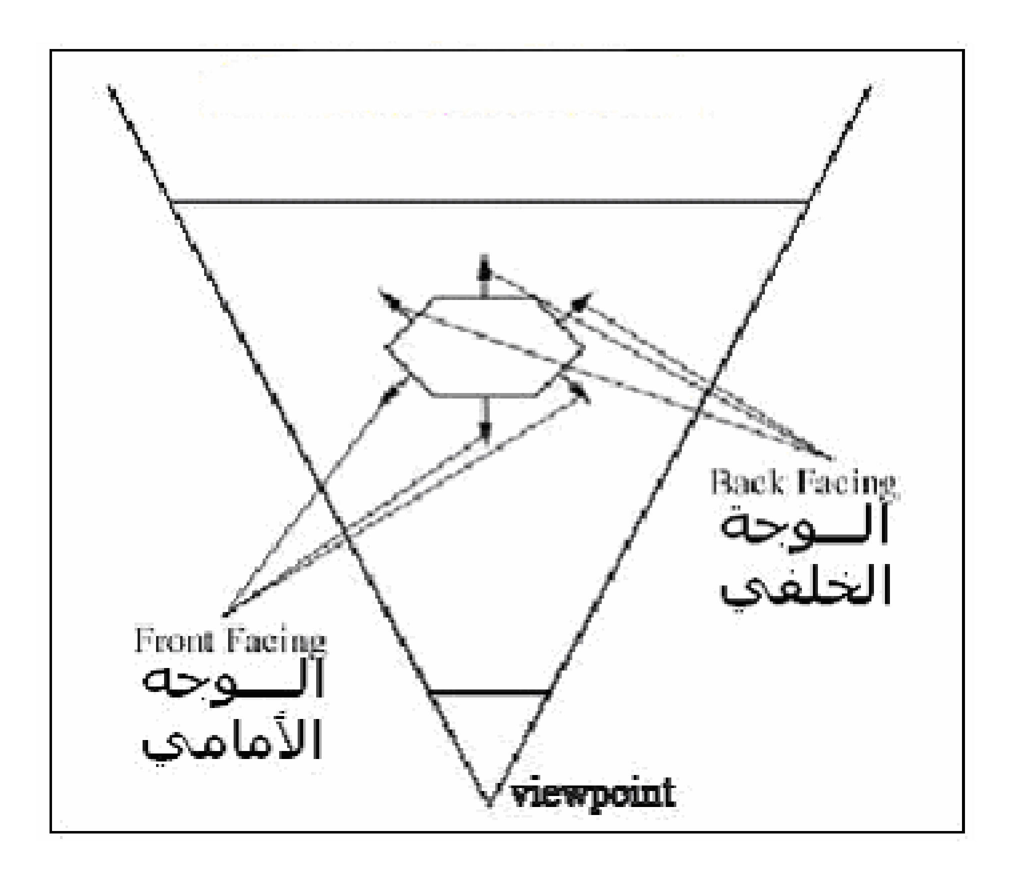


نمثل هذة الإحداثيات في الـ DirectX كالتالي:

كود:

```
device.Transform.View = Matrix.LookAtLH(new Vector3(0,0,-30), new Vector3(0,0,0), new Vector3(0,1,0));
```

سادساً :يجب علينا معرفة مصطلح الـ) Culling Face غربلة الأوجة), لكل جسم وجهان أمامي وخلفي, الأمامي وهو الوجة الذي يظهر أمامنا, الخلفي وهو الوجة الذي لا نراه حيث يحجبه الوجة الأمامي (أنظر إلى الشكل بالأسفل.)



توجد ثلاثة حالات للـ Culling وهي:

Cull.None : والتي تعني إلغاء عملية غربلة (إخفاء (الأوجة الخلفية بشكل كامل.

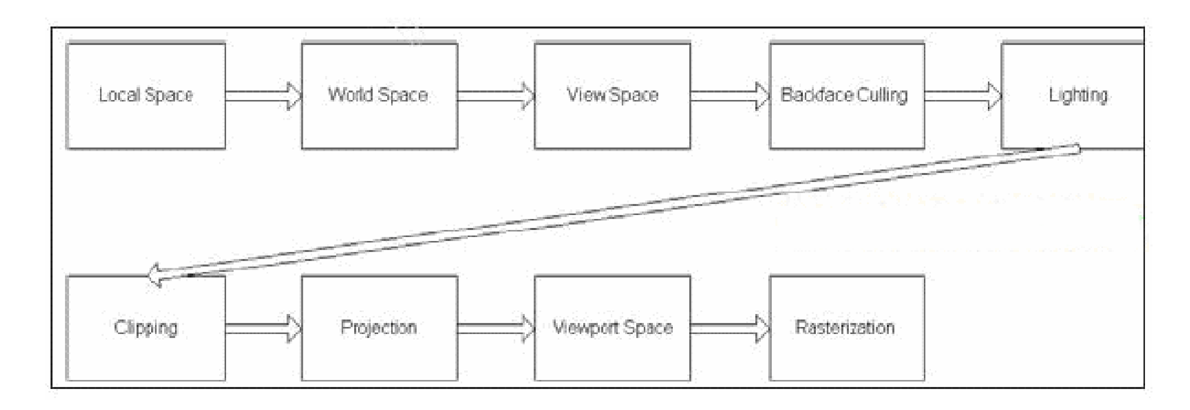
Cull.Clockwise : والتي تعني غربلة المثلثات التي تدور مع عقارب الساعة.

Cull.CounterClockwise : والتي تعني غربلة المثلثات التي تدور عكس عقارب الساعة (وهذا هي الحالة الإفتراضية.(

سابعاً: مصطلح الـ ,(Clipping) وهو يقوم على قص جميع الأجسام التي تقع في خارج حدود المنظور, أنظر إلى الشكل بالأسفل, فإنة سوف يقوم بي إظهار المثلث شكل A والجزء الأحمر من المثلث شكل ,B وسيقوم بقص الجزء ,C هذه العملية تريح الـ GPU و الـ CPU والذاكرة من عناء معالجة الرسومات التي تتوضع خارج منطقة المنظور.

ثامناً :موضوع الـ Lighting (الإضاءة) حيث سنتكلم عن بالتفصيل بالدروس القادمة, ويكفي إلى هذه المرحلة بأننا نضعه بي القيمة false

يمكننا تلحيص جميع النقاط بالأعلا ,بالشكل التالي:



الأن لنرى الكود بعد أن قمنا بإستخدام الـ
PerspectiveFovLH وLookAtLH و CullMode و Clipping و Lighting

كود:

```
using System;
using System.Drawing;
```

```
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace DirectX_Tutorial

{

public class WinForm : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container components = null;

public WinForm()
{

InitializeComponent();

this.SetStyle(ControlStyles.AllPaintingInWmPaint |
ControlStyles.Opaque, true);

}


public void InitializeDevice()

{

PresentParameters presentParams = new
PresentParameters();

presentParams.Windowed = true;

presentParams.SwapEffect = SwapEffect.Discard;

device = new Device(0, DeviceType.Hardware, this,
CreateFlags.SoftwareVertexProcessing, presentParams);

}


protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)

{
```

```
device.Transform.Projection =
Matrix.PerspectiveFovLH((float)Math.PI/4, this.Width/this.Height, -1f, 1f);

device.Transform.View = Matrix.LookAtLH(new Vector3(0,10,-
30), new Vector3(0,0,0), new Vector3(0,1,0));


device.RenderState.Lighting = false;

device.RenderState.CullMode = Cull.None ;
device.RenderState.Clipping = true;


CustomVertex.PositionColored[] vertices = new
CustomVertex.PositionColored[3];

vertices[0].SetPosition(new Vector3(0f, 0f, 0f));

vertices[0].Color = Color.Red.ToArgb();

vertices[1].SetPosition(new Vector3(10f, 0f, 0f));

vertices[1].Color = Color.Green.ToArgb();

vertices[2].SetPosition(new Vector3(5f, 10f, 0f));

vertices[2].Color = Color.Yellow.ToArgb();


device.Clear(ClearFlags.Target, Color.DarkSlateBlue , 1.0f, 0);


device.BeginScene();

device.VertexFormat =
CustomVertex.PositionColored.Format;

device.DrawUserPrimitives(PrimitiveType.TriangleList, 1,
vertices);

device.EndScene();


device.Present();


this.Invalidate();

}
```

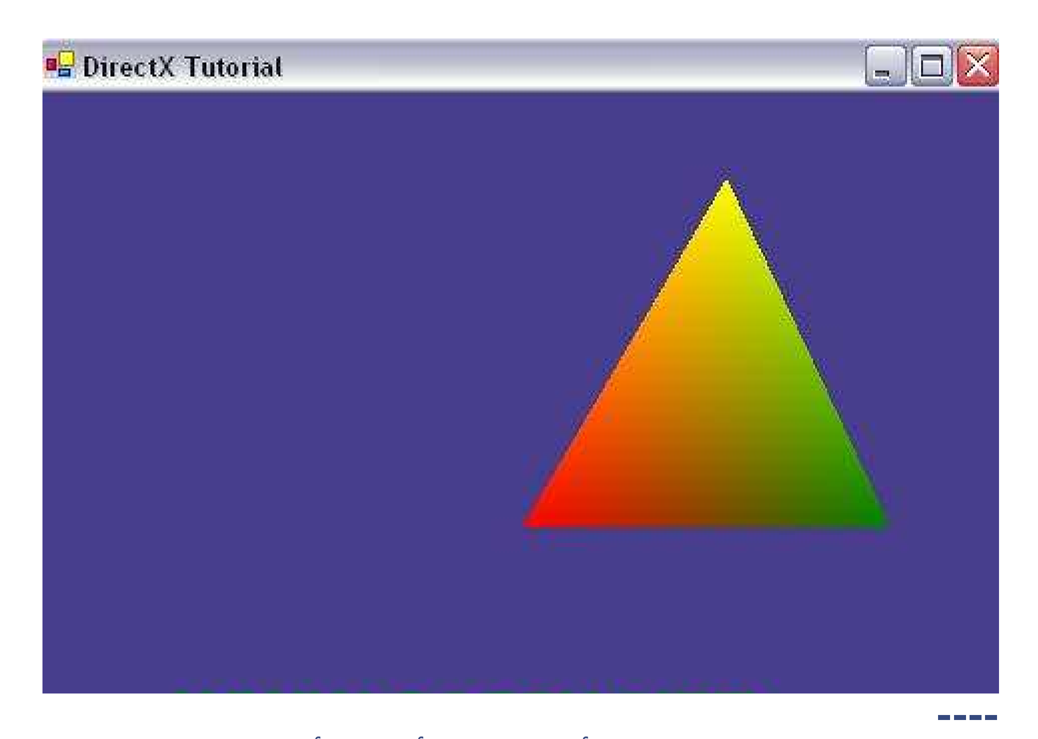
```
protected override void Dispose (bool disposing)
{
if (disposing)
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose(disposing);
}

private void InitializeComponent()
{
this.components = new System.ComponentModel.Container();
this.Size = new System.Drawing.Size(500,500);
this.Text = "DirectX Tutorial";
}

static void Main()
{
using (WinForm our_directx_form = new WinForm())
{
our_directx_form.InitializeDevice();
Application.Run(our_directx_form);
```

```
                }
            }
        }
    }
```

عند عمل run للكود بالأعلا فسيظهر الشكل بالأسفل:



إنتهى درس اليوم .... ولا أبالغ حين أقول بأن هذا الدرس يمثل حجر الأساس ... فيجب قراءته أكثر من مرة ليرسخ في الأذهان ...

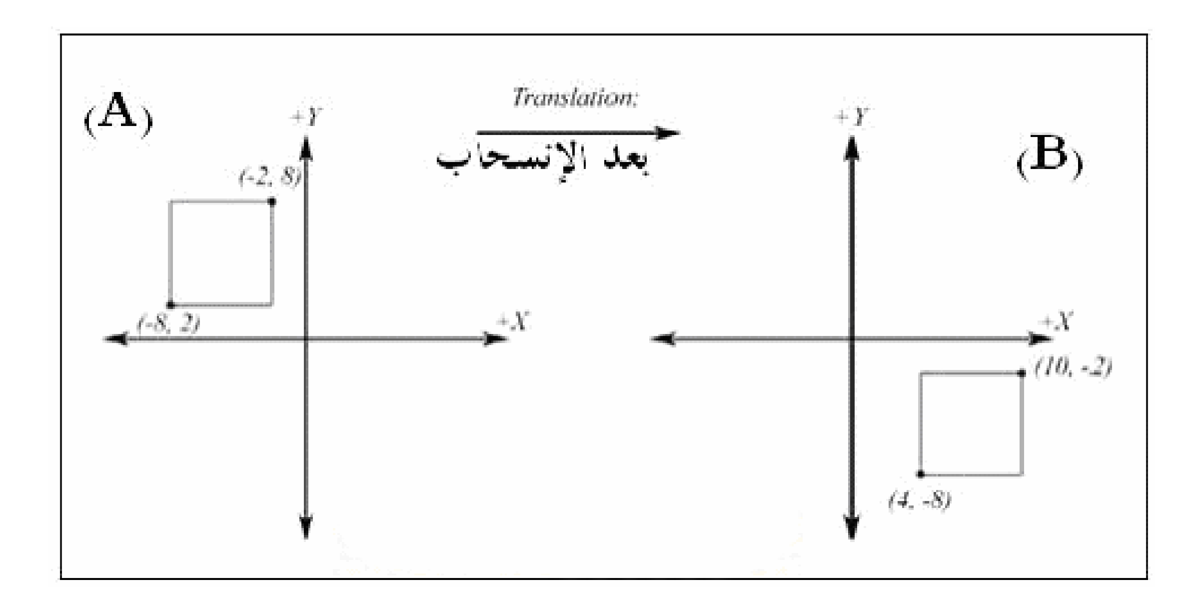
موعدنا مع الدرس التاسع (إن شاء اللّه...)

الدرس التاسع:

-الـ Translation
-الـ Rotation

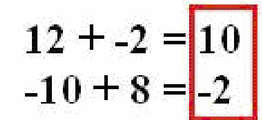
-الـ Translation

الـ Translation ويمثل: الإنسحاب عن الإحداثيات الرئيسية, تستخدم هذه الخاصية عند الرغبة بتحريك الكميرا أو الأجسام, فمثلاً عند الضغط من لوحة المفاتيح على الحرف R نجعل الجسم يتحرك على الجهه اليمنى أو على الحرف L ليتحرك على الجهه اليسرى , وهكذا.... لتوضيح الأمر فالنبدأ بمثال رياضي أنظر إلى الشكل بالأسفل:



نلاحظ في الشكل بالأعلا (A) والذي يحمل الإحداثية الأولي وهي:
x = ٢-
y = ٨
والإحداثية الثانية وهي:
x = ٨-
y = ٢

لنقل الآن أني أريد عمل إنسحاب للشكل A بمقدار ١٢ على المحور x, ومقدار -١٠ على المحورy



النقطة الأولى ستصبح (10,-2)

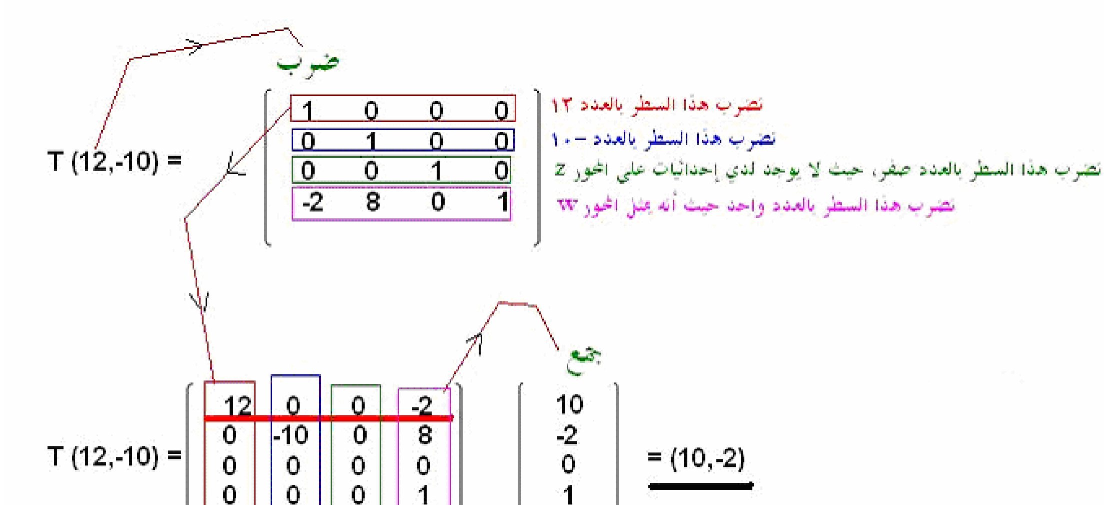
12 + -8 = 4
-10 + 2 = -8

النقطة الثانية ستصبح (4,-8)

ليعطينا في النهاية الشكل.(B)

في الحقيقة بي عالم الجرافيكس لا تتم العمليات هكذا وإنما تتم بواسطة المصفوفات, في بادئ الأمر كانت العمليات تتم عن طريق المصفوفات الثلاثية وهي ,(x,y,z) وبعدها إبتكروا المصفوفات الرباعية ٤ Matrix ٤xوهي (x,y,z,w) حيث يأخذ الإحداثي الرابع (w)دائماً القيمة واحد للحفاظ على توازن المصفوفة, في عالم الرياضيات لا يوجد أفضل من التطبيق العملي لتتوضح الفكرة أكثر: لنأخذ نفس المثال بالأعلا الشكل (A) والشكل .(B) ونجري عليه عملية الإنسحاب بإستخدام مصفوفة الإنسحاب.
تعطي مصفوفة الإنسحاب بي المعادلة التالية:

$$\mathbf{T}(\mathbf{p}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p_x & p_y & p_z & 1 \end{bmatrix}$$



$$T(12,-10) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -8 & 2 & 0 & 1 \\ X & Y & Z & W \end{bmatrix}$$

$$T(12,-10) = \begin{bmatrix} 12 & 0 & 0 & -8 \\ 0 & -10 & 0 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ -8 \\ 0 \\ 1 \end{bmatrix} = \underline{(4,-8)}$$

أراحنا الـ DirectX من هذا العناء والتحويلات الرياضية التي تراها بالأعلا وذلك بإستخدام الدالة ()Translation فيكفي وضع بداخل القوسين إحداثيات الإنسحاب وهو يتكفل بالباقي:

كود:

```
device.Transform.World = Matrix.Translation(-8,2,0)
```

حيث أن:

الـ : device وهو إسم الكائن لكرت الشاشة.

:Transform.worldأي أن التحويل سوف يحدث بالعالم الحقيقي

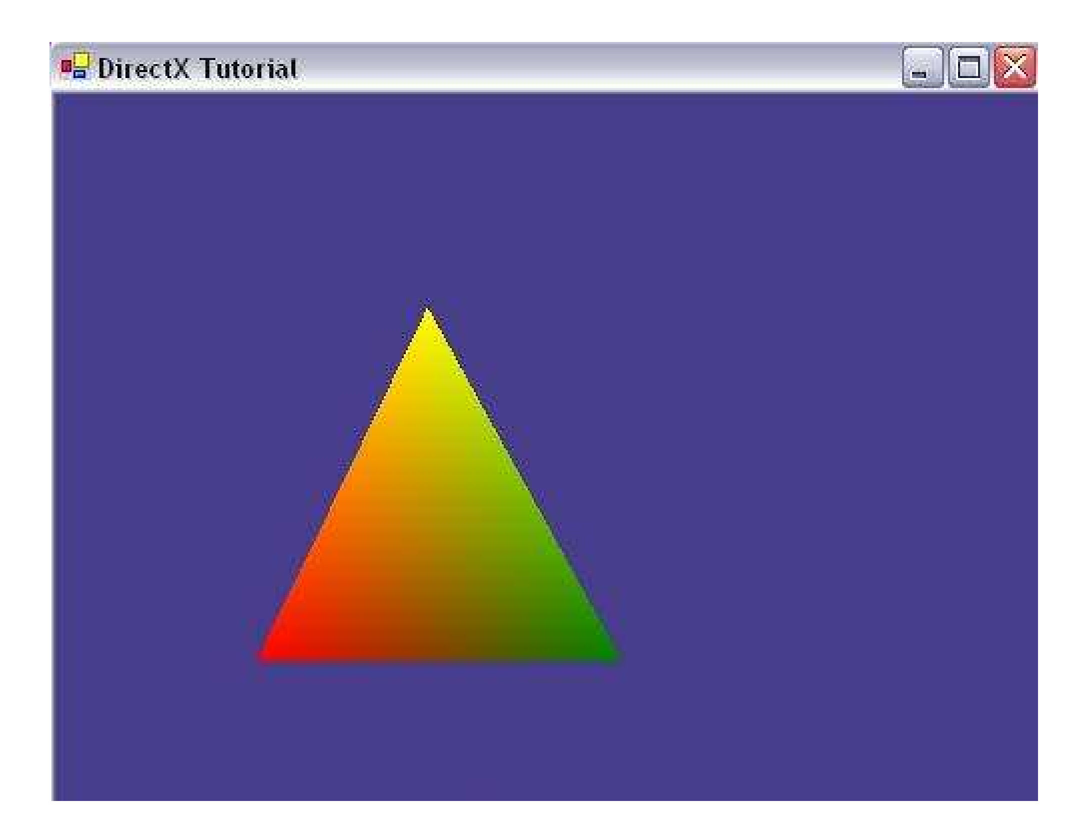
حيث سيكون ستكون الإحداثيات بالنسبة لباقي الأجسام.
وهي مصفوفة الإزاحة والتي تكلمنا عنهاMatrix.Translation بالأعلا.

الآن لنأخذ المثال (رسم المثلث) الذي تكلمنا عنه في الدرس الثامن , ونحدث عليه عملية الإنسحاب على المحور x بمقدار -٨ وعلى المحور y بمقدار -٣ وذلك بإضافة الجملة التالية بداخل الدالة OnPaint :

كود:

```
device.Transform.World = Matrix.Translation(-8,-3,0);
```

ليعطينا الشكل التالي:



نلاحظ بأن المثلث بعد أن كان في أعلا الجهه اليمنى, أصبح بعد عملية الإنسحاب كما في الشكل بالأعلا.

الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
```

```
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace DirectX_Tutorial

{

public class WinForm : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container components = null;

public WinForm()
{

InitializeComponent();

this.SetStyle(ControlStyles.AllPaintingInWmPaint |
ControlStyles.Opaque, true);

}


public void InitializeDevice()

{

PresentParameters presentParams = new
PresentParameters();

presentParams.Windowed = true;

presentParams.SwapEffect = SwapEffect.Discard;

device = new Device(0, DeviceType.Hardware, this,
CreateFlags.SoftwareVertexProcessing, presentParams);

}


protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)

{

device.Transform.Projection =
Matrix.PerspectiveFovLH((float)Math.PI/4, this.Width/this.Height, -1f, 1f);
```

```
device.Transform.View = Matrix.LookAtLH(new Vector3(0,10,-30), new Vector3(0,0,0), new Vector3(0,1,0));
device.Transform.World = Matrix.Translation(-8,-3,0);

device.RenderState.Lighting = false;

device.RenderState.CullMode = Cull.None ;
device.RenderState.Clipping = true;


CustomVertex.PositionColored[] vertices = new CustomVertex.PositionColored[3];

vertices[0].SetPosition(new Vector3(0f, 0f, 0f));

vertices[0].Color = Color.Red.ToArgb();

vertices[1].SetPosition(new Vector3(10f, 0f, 0f));

vertices[1].Color = Color.Green.ToArgb();

vertices[2].SetPosition(new Vector3(5f, 10f, 0f));

vertices[2].Color = Color.Yellow.ToArgb();


device.Clear(ClearFlags.Target, Color.DarkSlateBlue , 1.0f, 0);


device.BeginScene();

device.VertexFormat = CustomVertex.PositionColored.Format;

device.DrawUserPrimitives(PrimitiveType.TriangleList, 1, vertices);

device.EndScene();


device.Present();


this.Invalidate();

}
```

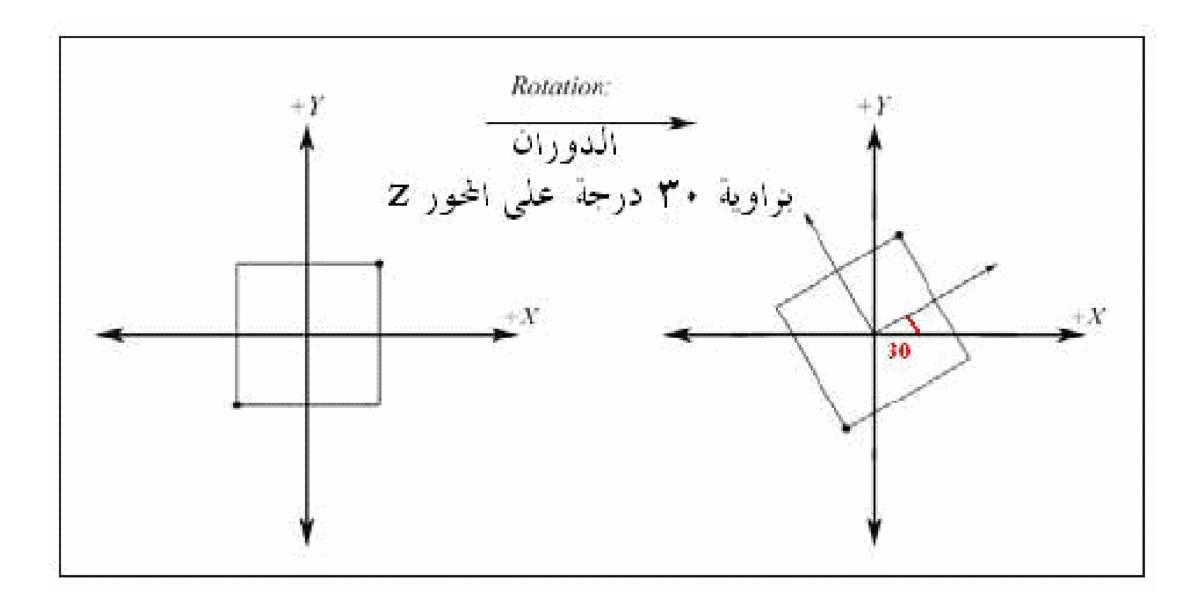
```
protected override void Dispose (bool disposing)
{
if (disposing)
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose(disposing);
}

private void InitializeComponent()
{
this.components = new System.ComponentModel.Container();
this.Size = new System.Drawing.Size(500,500);
this.Text = "DirectX Tutorial";
}
static void Main()
{
using (WinForm our_directx_form = new WinForm())
{
our_directx_form.InitializeDevice();
Application.Run(our_directx_form);
}
}
}
}
```

-الـRotation

ويمثل: تدوير الجسم بمقدار الزاوية حول المحور x و y وz

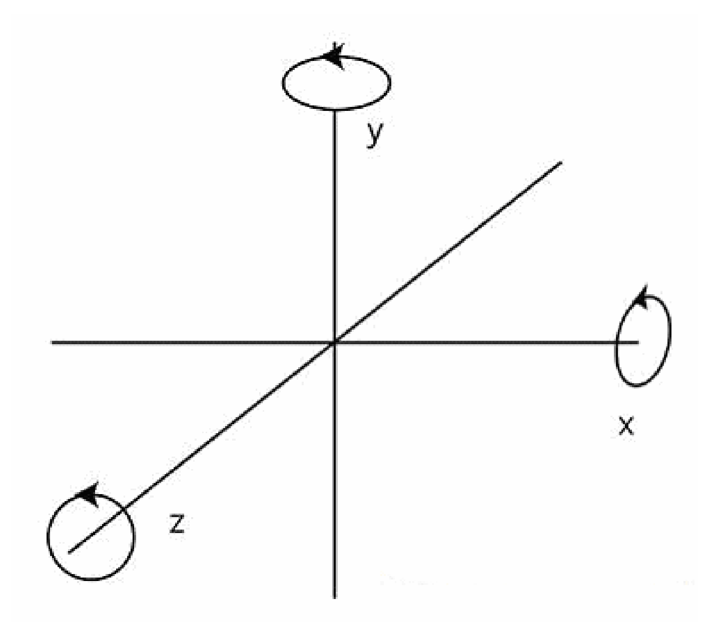
تخيل عملية الدوران كالتالي: لدي جسم على محور الإحداثيات, وأردنا أن نعمل (Rotation) لهذا الجسم بزاوية قدرها ٣٠ درجة, فسيظهر كما في الشكل بالأسفل:



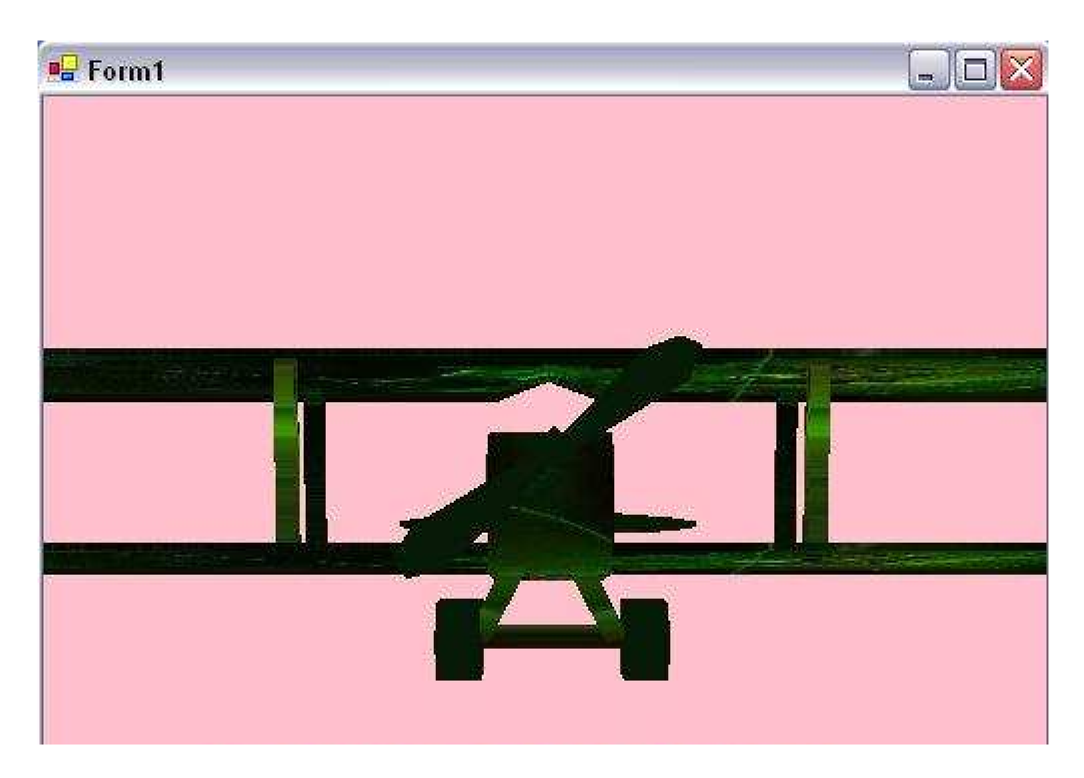
ملاحظات رياضية هامة :

أولاً: جميع درجات الزواية تكون بالراديان (Radian) وتستخدم الدالة التالية Geometry.DegreeToRadian من أجل التحويل من نظام الدرجات إلى الـ(Radian)

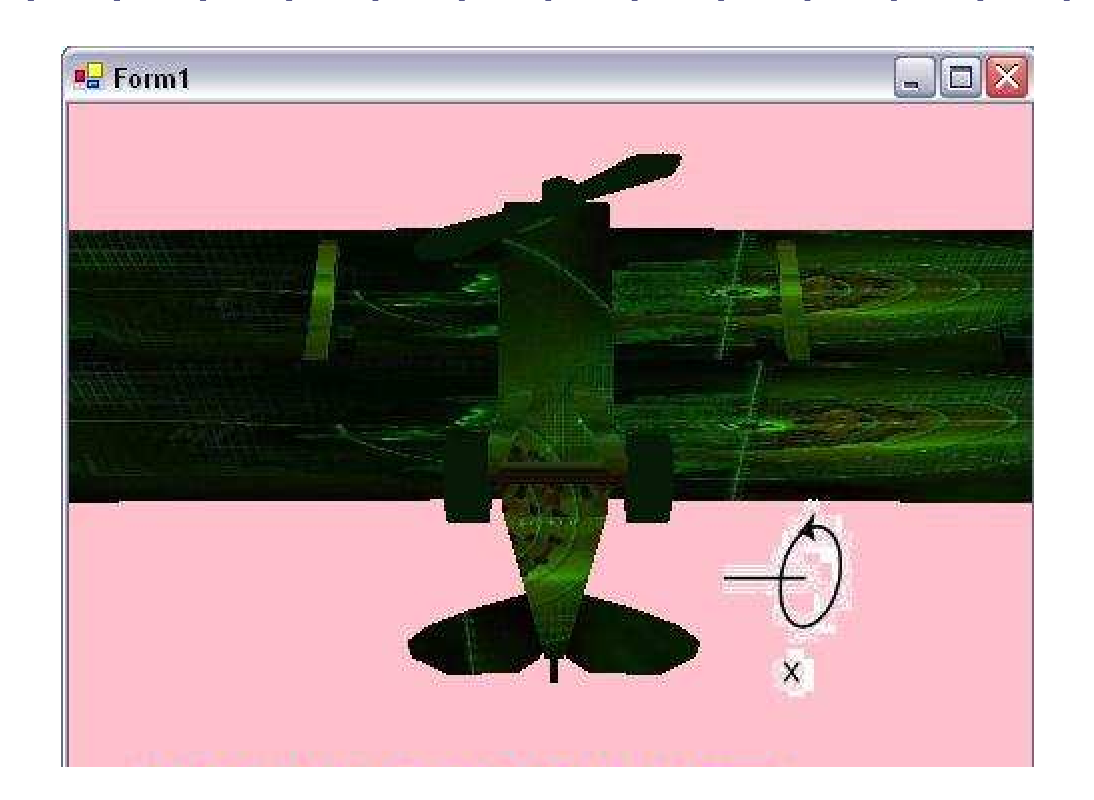
ثانياً: لكل محور طريقة في الدوران) أنظر إلى الشكل بالأسفل(



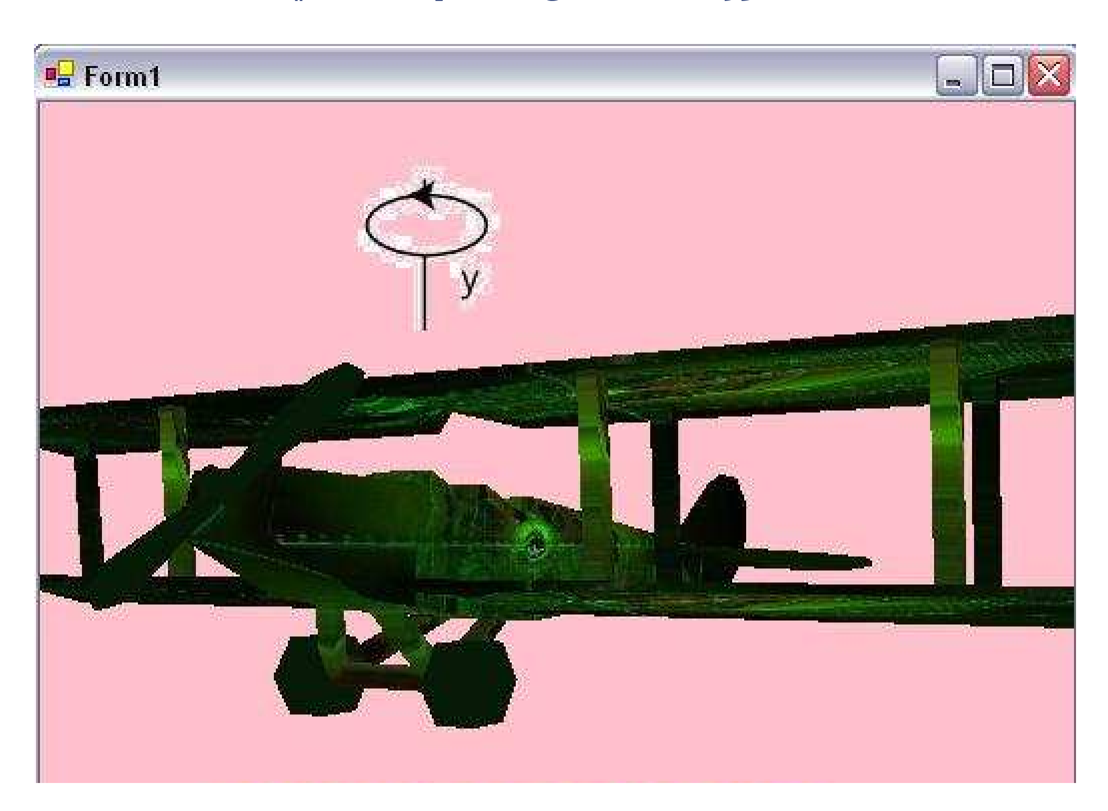
لنتخيل الأمر كالتالي: لنفرض أن لدي الشكل بالأسفل وهو طائرة مروحية:



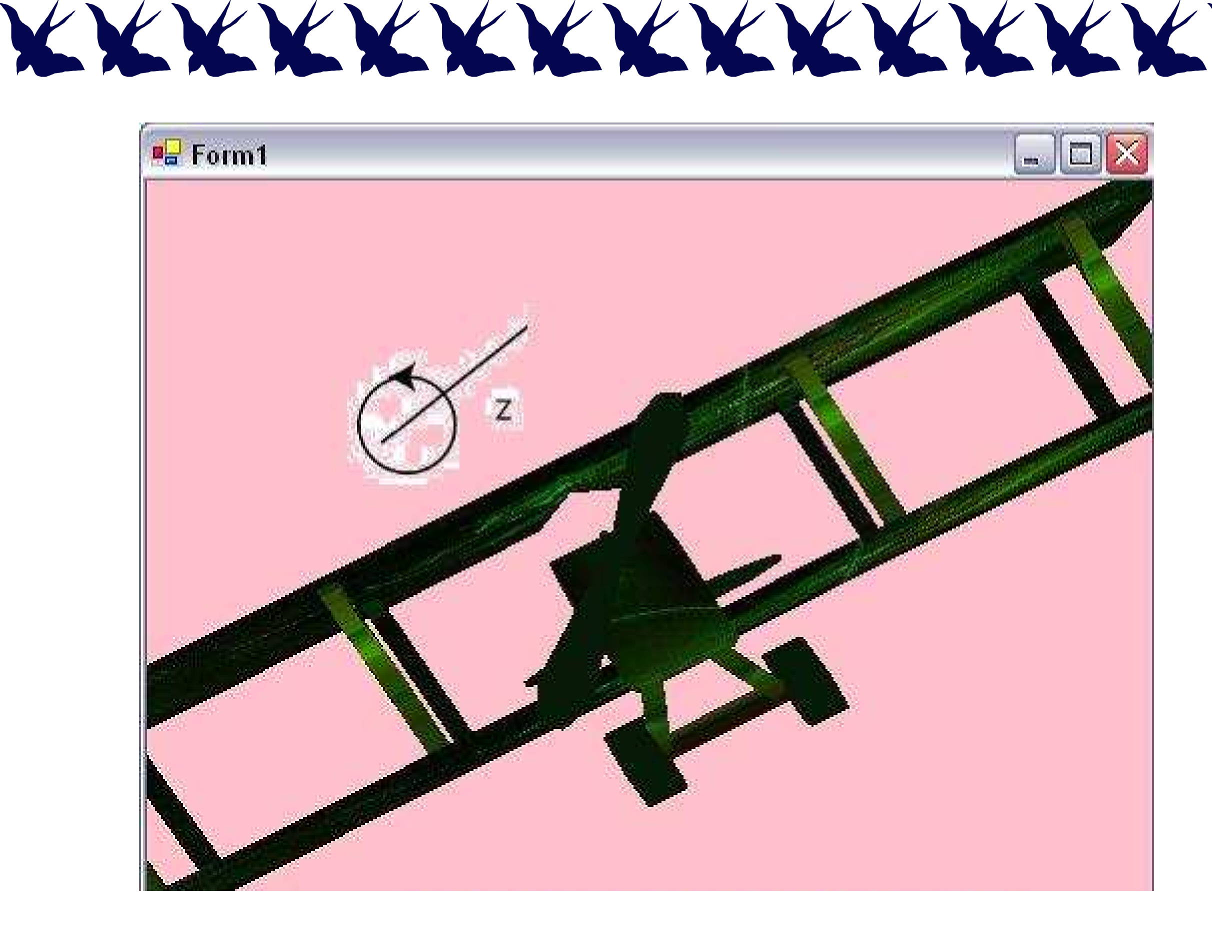
الآن سنقوم بعمل دوران لهذه الطائرة بزاوية قدرها ٣٠ درجة على المحور X, سيصبح شكلها كالتالي:



الآن سنقوم بعمل دوران لهذه الطائرة بزاوية قدرها ٣٠ درجة على المحور Y, سيصبح شكلها كالتالي:

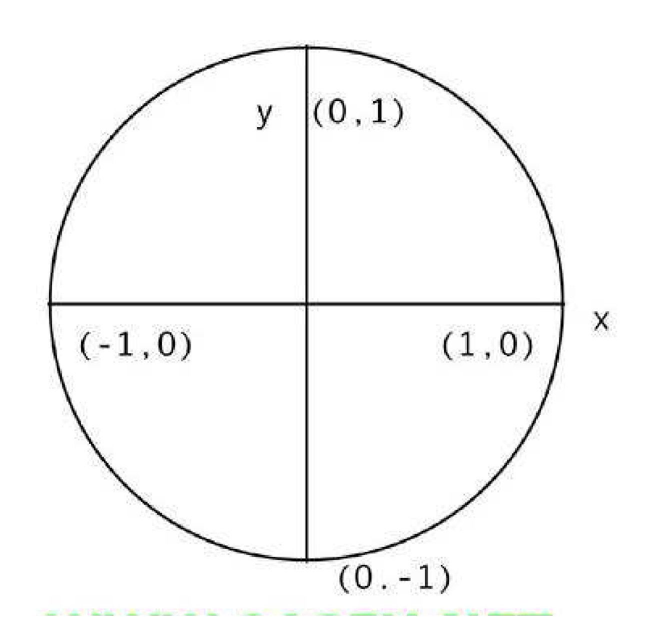


الآن سنقوم بعمل دوران لهذه الطائرة بزاوية قدرها ٣٠ درجة على المحور Z, سيصبح شكلها كالتالي:



ثالثاً: عند عمل دوران (Rotation) للجسم, فهذا يعني بأن الإحداثيات القديمة ستتغير بحسب قيمة الزاوية المعطاه.
رياضياً من أجل حساب هذه النقاط الجديدة, فنحن بحاجة إلى الإحداثيات القديمة و الـ Cos و الـ Sin لزاوية الدوران .

لنبدأ بمثال بسيط وهو دائرة الوحدة (وهي التي تكون جميع إحداثياتها تساوي واحد), كما في الشكل بالأسفل:

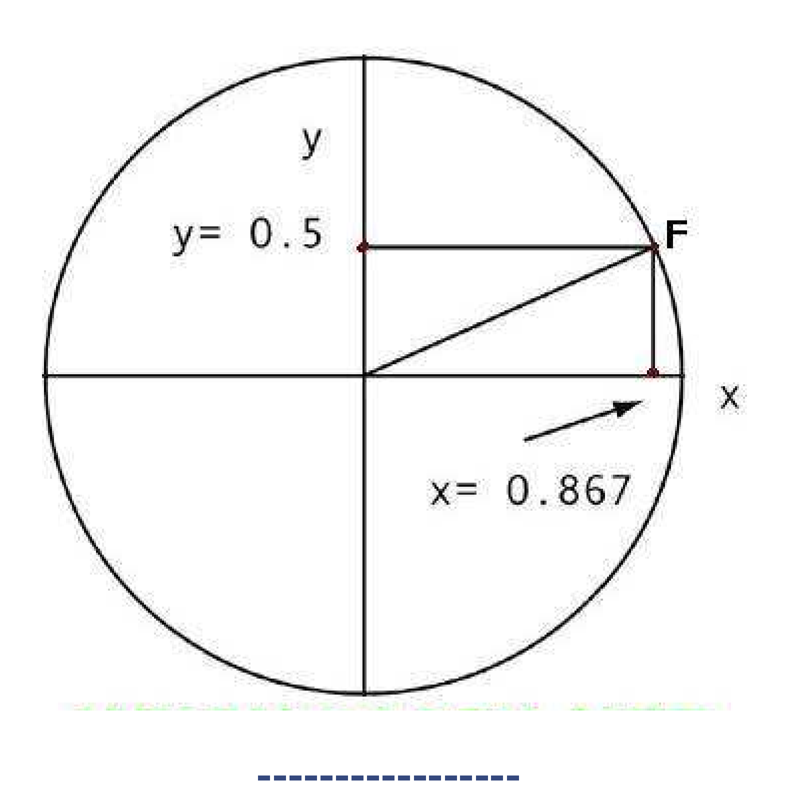


لنفرض أني أردت عمل دوران بزاوية ٣٠ درجة من نقطة الأصل كما في الشكل بالاسفل, عندها سنحتاج إلى معرفة الإحداثيات للـ X و الـ Y , نقوم بذلك عن طريق إسقاط عامودين من النقطة F, الأول على المحور X والثاني على المحور Y, ولحساب هاتين النقطتين تكون كالتالي:

X = Cos الـ٣٠ وتساوي : ٠,٨٦٧

Y = Sin الـ٣٠ وتساوي٠,٥ :

)أنظر إلى الشكل بالأسفل.(

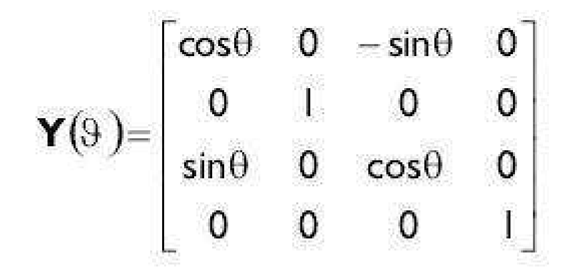


أما عند الـ DirectX فإنه يقوم بهذه العملية بإستخدام مصفوفة الدوران.(Matrix Rotation)

مصفوفة الدوران للمحور X هي:

$$\mathbf{X}(\vartheta)=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

مصفوفة الدوران للمحور Y هي:



مصفوفة الدوران للمحور Z هي:

$$Z(\vartheta)=\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

لنري كيف تعمل الـ Matrix سنأخذ نفس المثال في الدرس السابق وهو المثلث,ونطبق عليه عملية الدوران بزاوية ١٨٠ درجة على المحور X, لنرى الإحداثيات الجديدة للمثلث بعد عملية الدوران.

إحداثيات المثلث القديمة وهي:

كود:

```
vertices[0].SetPosition(new Vector3(0f, 0f, 0f));
vertices[1].SetPosition(new Vector3(10f, 0f, 0f));
vertices[2].SetPosition(new Vector3(5f, 10f, 0f));
```

مصفوفة الدوران للمحور X هي:

$$X(\vartheta)=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



النقطة الأولي :

كود:

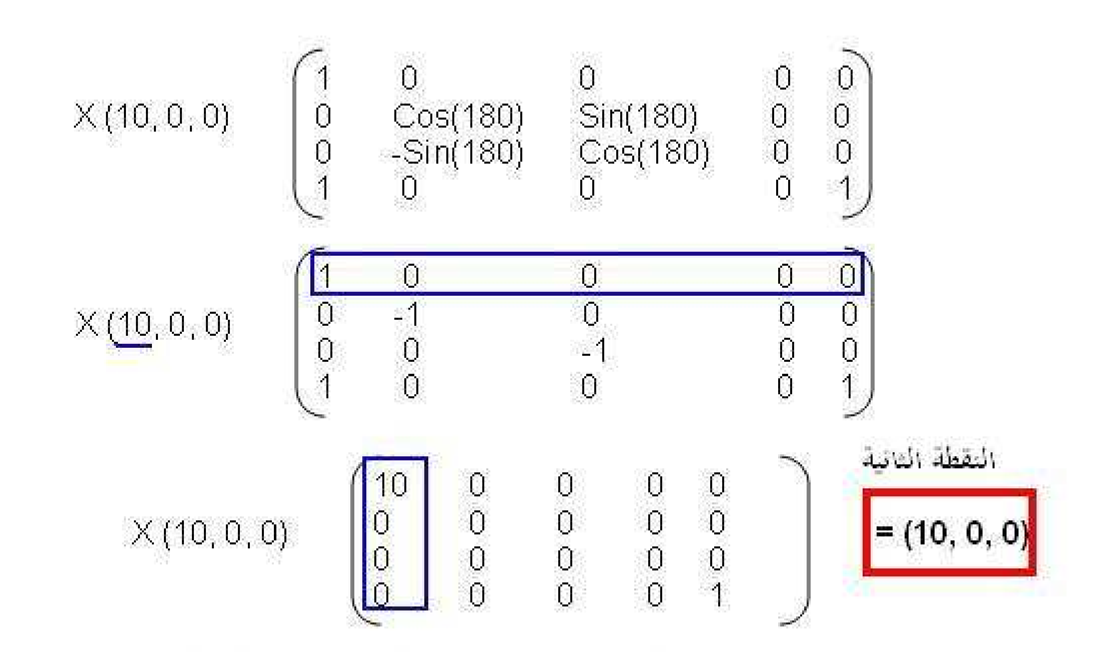
```
vertices[0].SetPosition(new Vector3(0f, 0f, 0f));
```

بما أن المحاور الثلاثة تساوي صفر, إذن فلا يوجد داعي لحساب الإختلاف لأن النتيجة ستبقي كما هي (٠،٠،٠).

النقطة الثانية :

كود:

```
vertices[1].SetPosition(new Vector3(10f, 0f, 0f));
```



النقطة الثالثة:

كود:

```
vertices[2].SetPosition(new Vector3(5f, 10f, 0f));
```

X (5, 10, 0)
1 0 0 0 0
0 Cos(180) Sin(180) 0 0
0 -Sin(180) Cos(180) 0 0
1 0 0 0 1

X (5, 10, 0)
1 0 0 0 0
0 -1 0 0 0
0 0 -1 0 0
1 0 0 0 1

X (5, 10, 0)
5 0 0 0 0
0 -10 0 0 0
0 0 0 0 0
0 0 0 0 1

= (5, -10, 0)
النقطة الثالثة



إحداثيات المثلث الجديدة بعد عملية الدوران بزاوية ١٨٠ درجة:

كود:

```
vertices[0].SetPosition(new Vector3(0f, 0f, 0f));
vertices[1].SetPosition(new Vector3(10f, 0f, 0f));
vertices[2].SetPosition(new Vector3(5f, -10f, 0f));
```

دعني الآن أطلعك على سر من كل ما سبق أعلاه يكفي أن تعلم التالي: يوجد في الـ DirectX دالة تريحنا من عناء الحسابات التي بالأعلا وهي:

دالة الدوران على المحورX

كود:

```
device.Transform.World = Matrix.RotationX();
```

دالة الدوران على المحورY

كود:

```
device.Transform.World = Matrix.RotationY();
```

دالة الدوران على المحورZ

كود:

```
device.Transform.World = Matrix.RotationZ();
```

دالة الدوران على المحاور الثلاثة(X,Y,Z)

كود:

```
device.Transform.World = Matrix. RotationAxis ();
```

لنقوم بحل المثال بالأعلا وهو الدوران بزاوية ١٨٠ درجة على المحور X, الحل بغاية البساطة وكما يقوم أستاذنا دائماً As a pies of Cake فيكفي إضافة هذا السطر في الدالة OnPaint لتتكفل بالباقي.

كود:

```
device.Transform.World = Matrix.RotationX(Geometry.DegreeToRadian
(180));
```

الكود كامل:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

namespace DirectX_Tutorial

{

public class WinForm : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container components = null;
private float angle ;
public WinForm()
{

InitializeComponent();

this.SetStyle(ControlStyles.AllPaintingInWmPaint |
ControlStyles.Opaque, true);

}


public void InitializeDevice()

{

PresentParameters presentParams = new
PresentParameters();

presentParams.Windowed = true;

presentParams.SwapEffect = SwapEffect.Discard;

device = new Device(0, DeviceType.Hardware, this,
CreateFlags.SoftwareVertexProcessing, presentParams);
```

```
}

protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)

{

device.Transform.Projection =
Matrix.PerspectiveFovLH((float)Math.PI/4, this.Width/this.Height, -1f, 1f);

device.Transform.View = Matrix.LookAtLH(new Vector3(0,0,-
30), new Vector3(0,0,0), new Vector3(0,1,0));

device.Transform.World =
Matrix.RotationX  (Geometry.DegreeToRadian (180));

device.RenderState.Lighting = false;

device.RenderState.CullMode = Cull.None ;
device.RenderState.Clipping = true;

CustomVertex.PositionColored[] vertices = new
CustomVertex.PositionColored[3];

vertices[0].SetPosition(new Vector3(0f, 0f, 0f));

vertices[0].Color = Color.Red.ToArgb();

vertices[1].SetPosition(new Vector3(10f, 0f, 0f));

vertices[1].Color = Color.Green.ToArgb();

vertices[2].SetPosition(new Vector3(5f, 10f, 0f));

vertices[2].Color = Color.Yellow.ToArgb();

device.Clear(ClearFlags.Target, Color.DarkSlateBlue , 1.0f, 0);

device.BeginScene();
```

```
device.VertexFormat =
CustomVertex.PositionColored.Format;

device.DrawUserPrimitives(PrimitiveType.TriangleList, 1,
vertices);

device.EndScene();


device.Present();


this.Invalidate();
angle += 0.05f;

}


protected override void Dispose (bool disposing)

{

if (disposing)

{

if (components != null)

{

components.Dispose();

}

}

base.Dispose(disposing);

}


private void InitializeComponent()

{

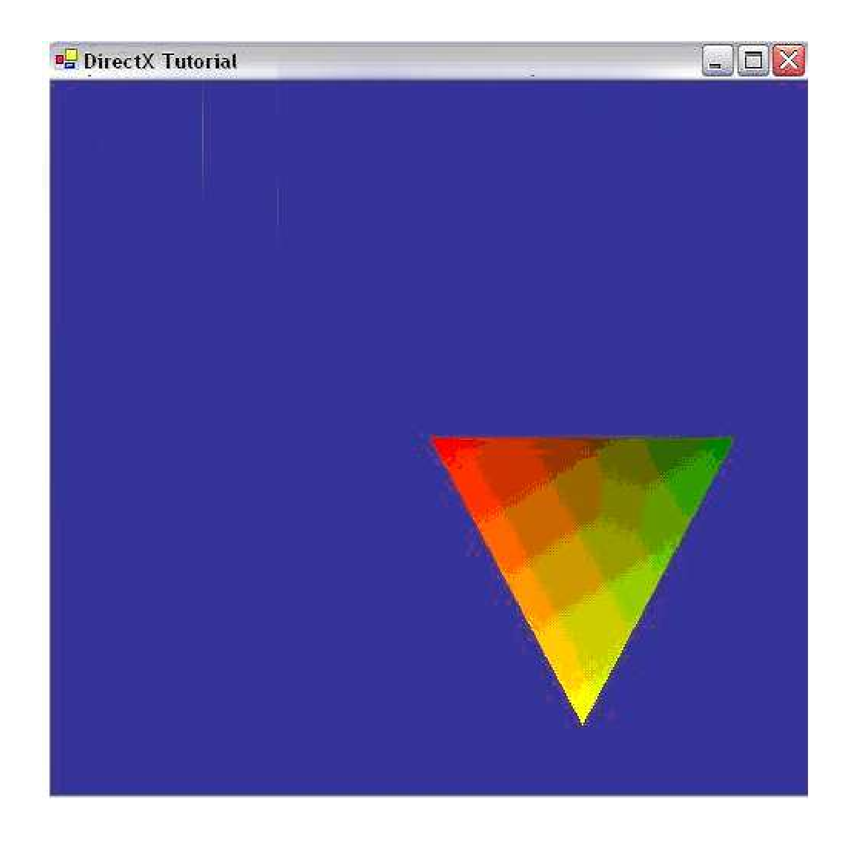
this.components = new System.ComponentModel.Container();

this.Size = new System.Drawing.Size(500,500);
```

```
this.Text = "DirectX Tutorial";
}

static void Main()
{
using (WinForm our_directx_form = new WinForm())
{
our_directx_form.InitializeDevice();
Application.Run(our_directx_form);
}
}
}
}
```

عند عمل RUN للكود بالأعلا فسيظهر الشكل التالي:



ما رأيك الآن أن نضيف بأن نجعل الجسم يدور بزاوية تزايدية ..
لعمل ذلك:
أولاً :نقوم بالتصريح عن متغير ولنعطية الإسم angle كالتالي:

كود:

```
private float angle ;
```

ثانياً :نقوم بإعطاء الزاوية قيمة تزايدية ولتكن ٠,٠٥ , فكلما زادت هذه القيمة زاد بالمقابل سرعة دوران الجسم.

كود:

```
angle += 0.05f;
```

ثالثاً :إعطاء دالة الدوران قيمة الـ angle كالتالي:

كود:

```
device.Transform.World = Matrix.RotationX  (angle);
```

الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace DirectX_Tutorial

{

public class WinForm : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container components = null;
private float angle ;
public WinForm()
{

InitializeComponent();

this.SetStyle(ControlStyles.AllPaintingInWmPaint |
ControlStyles.Opaque, true);
```

```
}

public void InitializeDevice()

{

PresentParameters presentParams = new
PresentParameters();

presentParams.Windowed = true;

presentParams.SwapEffect = SwapEffect.Discard;

device = new Device(0, DeviceType.Hardware, this,
CreateFlags.SoftwareVertexProcessing, presentParams);

}

protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)

{

device.Transform.Projection =
Matrix.PerspectiveFovLH((float)Math.PI/4, this.Width/this.Height, -1f, 1f);

device.Transform.View = Matrix.LookAtLH(new Vector3(0,0,-
30), new Vector3(0,0,0), new Vector3(0,1,0));

device.Transform.World =
Matrix.RotationX  (angle);

device.RenderState.Lighting = false;

device.RenderState.CullMode = Cull.None ;
device.RenderState.Clipping = true;

CustomVertex.PositionColored[] vertices = new
CustomVertex.PositionColored[3];

vertices[0].SetPosition(new Vector3(0f, 0f, 0f));
```

```
vertices[0].Color = Color.Red.ToArgb();

vertices[1].SetPosition(new Vector3(10f, 0f, 0f));

vertices[1].Color = Color.Green.ToArgb();

vertices[2].SetPosition(new Vector3(5f, 10f, 0f));

vertices[2].Color = Color.Yellow.ToArgb();


device.Clear(ClearFlags.Target, Color.DarkSlateBlue , 1.0f, 0);


device.BeginScene();

device.VertexFormat =
CustomVertex.PositionColored.Format;

device.DrawUserPrimitives(PrimitiveType.TriangleList, 1,
vertices);

device.EndScene();



device.Present();


this.Invalidate();
        angle += 0.05f;

}


protected override void Dispose (bool disposing)

{

if (disposing)

{

if (components != null)

{

components.Dispose();
```

```
}
}
base.Dispose(disposing);
}

private void InitializeComponent()
{
this.components = new System.ComponentModel.Container();
this.Size = new System.Drawing.Size(500,500);
this.Text = "DirectX Tutorial";
}

static void Main()
{
using (WinForm our_directx_form = new WinForm())
{
our_directx_form.InitializeDevice();
Application.Run(our_directx_form);
}
}
}
}
```

يمكنك أيضاً إعطاء زوايا للمحاور الثلاثة بالإضافة إلى القيمة تزايدية: الكود:

كود:

```
using System;
using System.Drawing;
```

```
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace DirectX_Tutorial

{

public class WinForm : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container components = null;
private float angle ;
public WinForm()
{

InitializeComponent();

this.SetStyle(ControlStyles.AllPaintingInWmPaint |
ControlStyles.Opaque, true);

}


public void InitializeDevice()

{

PresentParameters presentParams = new
PresentParameters();

presentParams.Windowed = true;

presentParams.SwapEffect = SwapEffect.Discard;

device = new Device(0, DeviceType.Hardware, this,
CreateFlags.SoftwareVertexProcessing, presentParams);

}


protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)

{
```

```
device.Transform.Projection =
Matrix.PerspectiveFovLH((float)Math.PI/4, this.Width/this.Height, -1f, 1f);

device.Transform.View = Matrix.LookAtLH(new Vector3(0,0,-
30), new Vector3(0,0,0), new Vector3(0,1,0));

device.Transform.World =
Matrix.RotationAxis  (new Vector3 (2,2,2),angle);


device.RenderState.Lighting = false;

device.RenderState.CullMode = Cull.None ;
device.RenderState.Clipping = true;


CustomVertex.PositionColored[] vertices = new
CustomVertex.PositionColored[3];

vertices[0].SetPosition(new Vector3(0f, 0f, 0f));

vertices[0].Color = Color.Red.ToArgb();

vertices[1].SetPosition(new Vector3(10f, 0f, 0f));

vertices[1].Color = Color.Green.ToArgb();

vertices[2].SetPosition(new Vector3(5f, 10f, 0f));

vertices[2].Color = Color.Yellow.ToArgb();


device.Clear(ClearFlags.Target, Color.DarkSlateBlue , 1.0f, 0);


device.BeginScene();

device.VertexFormat =
CustomVertex.PositionColored.Format;

device.DrawUserPrimitives(PrimitiveType.TriangleList, 1,
vertices);

device.EndScene();
```

```
device.Present();

this.Invalidate();
        angle += 0.05f;

}

protected override void Dispose (bool disposing)

{

if (disposing)

{

if (components != null)

{

components.Dispose();

}

}

base.Dispose(disposing);

}

private void InitializeComponent()

{

this.components = new System.ComponentModel.Container();

this.Size = new System.Drawing.Size(500,500);

this.Text = "DirectX Tutorial";

}

static void Main()
```

```
{
using (WinForm our_directx_form = new WinForm())
{
our_directx_form.InitializeDevice();
Application.Run(our_directx_form);
}
}
}
}
```

هناك أيضاً طريقة أخرى من أجل عمل قيمة تزايدية للزاوية وهي إستخدام الخاصية
TickCountوالتي تبلغ قيمتها ١٥milliseconds

كود:

```
Matrix.RotationX((System.Environment.TickCount))
```

الكود بالكامل:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

namespace DirectX_Tutorial
{
public class WinForm : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container components = null;
//                                private float angle ;
public WinForm()
```

```
{
InitializeComponent();
this.SetStyle(ControlStyles.AllPaintingInWmPaint |
ControlStyles.Opaque, true);
}

public void InitializeDevice()
{
PresentParameters presentParams = new
PresentParameters();
presentParams.Windowed = true;
presentParams.SwapEffect = SwapEffect.Discard;
device = new Device(0, DeviceType.Hardware, this,
CreateFlags.SoftwareVertexProcessing, presentParams);
}

protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)
{
device.Transform.Projection =
Matrix.PerspectiveFovLH((float)Math.PI/4, this.Width/this.Height, -1f, 1f);
device.Transform.View = Matrix.LookAtLH(new Vector3(0,0,-
30), new Vector3(0,0,0), new Vector3(0,1,0));
device.Transform.World =
Matrix.RotationX((System.Environment.TickCount / 450.0f));

device.RenderState.Lighting = false;
```

```
device.RenderState.CullMode = Cull.None ;
device.RenderState.Clipping = true;

CustomVertex.PositionColored[] vertices = new
CustomVertex.PositionColored[3];

vertices[0].SetPosition(new Vector3(0f, 0f, 0f));

vertices[0].Color = Color.Red.ToArgb();

vertices[1].SetPosition(new Vector3(10f, 0f, 0f));

vertices[1].Color = Color.Green.ToArgb();

vertices[2].SetPosition(new Vector3(5f, 10f, 0f));

vertices[2].Color = Color.Yellow.ToArgb();

device.Clear(ClearFlags.Target, Color.DarkSlateBlue , 1.0f, 0);

device.BeginScene();

device.VertexFormat =
CustomVertex.PositionColored.Format;

device.DrawUserPrimitives(PrimitiveType.TriangleList, 1,
vertices);

device.EndScene();

device.Present();

this.Invalidate();
//                    angle += 0.05f;

}

protected override void Dispose (bool disposing)

{

if (disposing)
```

```
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose(disposing);
}

private void InitializeComponent()
{
this.components = new System.ComponentModel.Container();
this.Size = new System.Drawing.Size(500,500);
this.Text = "DirectX Tutorial";
}


static void Main()
{
using (WinForm our_directx_form = new WinForm())
{
our_directx_form.InitializeDevice();
Application.Run(our_directx_form);
}
}
}
}
```



الدرس العاشر:

-الـScaling
-جمع الـ Matrix للـ Translation و الـ Rotation و الـ Scaling

-الـScaling

الـ Scaling ويمثل: تكبير وتصغير المجسم في الإتجاهات x و y و. z

لنأخذ المثال التالي: لنفرض أن لدي جسم يحمل النقاط التاليه:
النقطة الأولي : المحور X يساوي ٤ -------- والمحور Y يساوي ٤
النقطة الثانية : المحور X يساوي -٤ -------- والمحور Y يساوي -٤
كما في الشكل(A)

لنقل الآن أني أريد عمل Scaling على الشكل (A), بحيث تكون على المحور X نصف) ٢/١ وعلى المحور Y تساوي ٢.
لنحل هذه المسألة كما يفعل الـ DirectX حيث سنستخدم هذه المصفوفة:(Matrix)

$$\mathbf{S(q)}=\begin{bmatrix} q_x & 0 & 0 & 0 \\ 0 & q_y & 0 & 0 \\ 0 & 0 & q_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$S(1/2, 2, 0) \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$S(1/2, 2, 0) \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

النقطة الأولى

= (2, 8, 0)

أنظر إلى الشكل (B)

$$) \begin{pmatrix} -4 & 0 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$) \begin{pmatrix} -2 & 0 & 0 & 0 & 0 \\ 0 & -8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

النقطة الثانية

= (-2, -8, 0)

أنظر إلى الشكل (B)

حسناً الآن إذا قلت لك أرجع لي الجسم إلى حالته الأولى قبل إحداث عملية الـ Scaling عليه (أي إرجاع الجسم من الشكل B إلى اشكل A).

عندئذن نستخدم معكوس المصفوفة (Matrix) لتصبح كالتالي:

$$\mathbf{S}^{-1} = \mathbf{S}\left(\frac{1}{q_x}, \frac{1}{q_y}, \frac{1}{q_z}\right) = \begin{bmatrix} \frac{1}{q_x} & 0 & 0 & 0 \\ 0 & \frac{1}{q_y} & 0 & 0 \\ 0 & 0 & \frac{1}{q_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$S\left(\frac{1}{1/2}, \frac{1}{2}, 0\right) \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

النقطة الأولى

$$S(2, 1/2, 0) \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} = (4, 4, 0)$$

$$S\left(\frac{1}{1/2}, \frac{1}{2}, 0\right) \begin{pmatrix} -2 & 0 & 0 & 0 & 0 \\ 0 & -8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

النقطة الثانية

$$S(2, 1/2, 0) \begin{pmatrix} -4 & 0 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} = (-4, -4, 0)$$

يقوم الـ DirectX بكل هذه التحويلات بإستخدام الدالة Matrix.Scaling()حيث نقوم بإضافة هذه الداله بداخل الـ OnPaint, لنأخذ مثال المثلث ونطبق عليه نظرية الـ Scaling بحيث يكون على المحور x يساوي ١ وعلى المحور y بساوي ٢ وعلى المحور z

يساوي ٠

كود:

```
device.Transform.World = Matrix.Scaling (1,2,0);
```

الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace DirectX_Tutorial

{

public class WinForm : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container components = null;
private float angle ;
public WinForm()
{

InitializeComponent();

this.SetStyle(ControlStyles.AllPaintingInWmPaint |
ControlStyles.Opaque, true);

}


public void InitializeDevice()

{

PresentParameters presentParams = new
PresentParameters();

presentParams.Windowed = true;
```

```
presentParams.SwapEffect = SwapEffect.Discard;

device = new Device(0, DeviceType.Hardware, this,
CreateFlags.SoftwareVertexProcessing, presentParams);

}


protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)

{

device.Transform.Projection =
Matrix.PerspectiveFovLH((float)Math.PI/4, this.Width/this.Height, -1f, 1f);

device.Transform.View = Matrix.LookAtLH(new Vector3(0,0,-
30), new Vector3(0,0,0), new Vector3(0,1,0));

device.Transform.World =
Matrix.Scaling (1,2,0);



device.RenderState.Lighting = false;

device.RenderState.CullMode = Cull.None ;
device.RenderState.Clipping = true;


CustomVertex.PositionColored[] vertices = new
CustomVertex.PositionColored[3];

vertices[0].SetPosition(new Vector3(0f, 0f, 0f));

vertices[0].Color = Color.Red.ToArgb();

vertices[1].SetPosition(new Vector3(10f, 0f, 0f));

vertices[1].Color = Color.Green.ToArgb();

vertices[2].SetPosition(new Vector3(5f, 10f, 0f));

vertices[2].Color = Color.Yellow.ToArgb();


device.Clear(ClearFlags.Target, Color.DarkSlateBlue , 1.0f, 0);
```

```
device.BeginScene();

device.VertexFormat =
CustomVertex.PositionColored.Format;

device.DrawUserPrimitives(PrimitiveType.TriangleList, 1,
vertices);

device.EndScene();


device.Present();


this.Invalidate();
angle += 0.05f;

}

protected override void Dispose (bool disposing)

{

if (disposing)

{

if (components != null)

{

components.Dispose();

}

}

base.Dispose(disposing);

}

private void InitializeComponent()

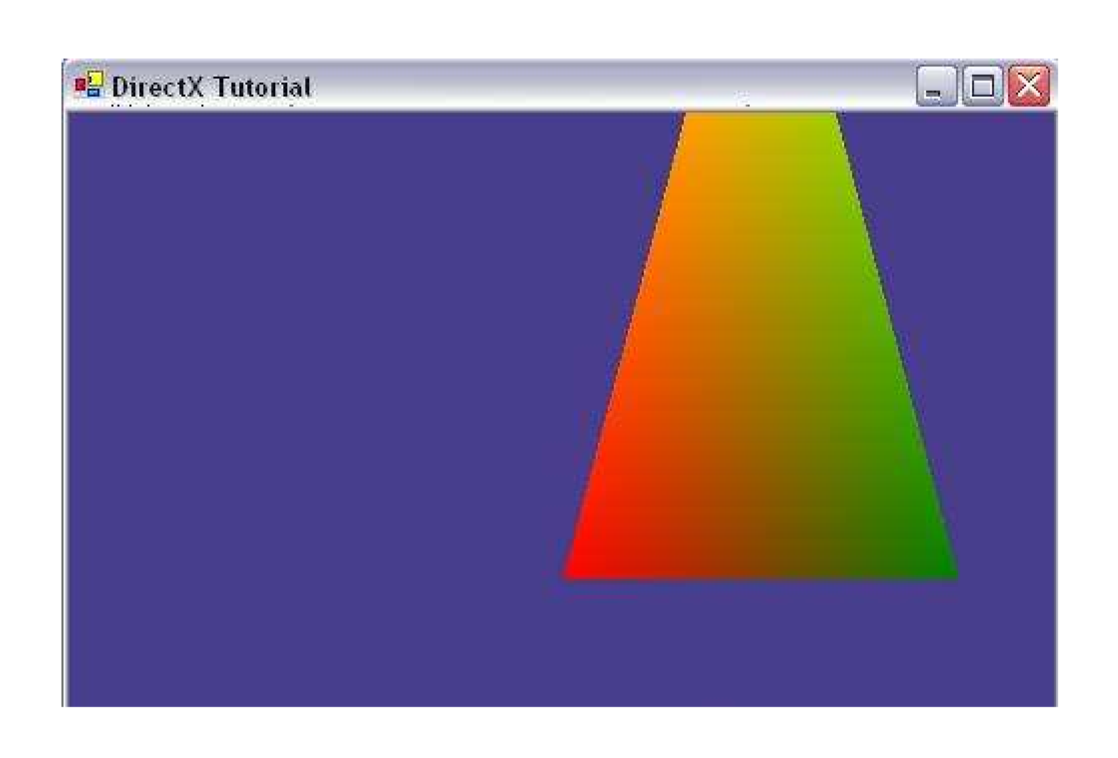
{

this.components = new System.ComponentModel.Container();
```

```
this.Size = new System.Drawing.Size(500,500);
this.Text = "DirectX Tutorial";
}

static void Main()
{
using (WinForm our_directx_form = new WinForm())
{
our_directx_form.InitializeDevice();
Application.Run(our_directx_form);
}
}
}
}
```

عند عمل RUN لهذا الكود فسيعطينا الشكل التالي:



-جمع الـ Matrix للـ Translation و الـ Rotation و الـ Scaling

ومن أجل فهم هذه العملية سنتكلم عن كيفية جمع وطرح وضرب المصفوفات(Matrix)

نبدأ بعملية الجمع:
لنفرض بأن لدي المصفوفة (A) والمصفوفة:(B)

$$A=\begin{bmatrix}1 & 5\\ -2 & 3\end{bmatrix} \quad B=\begin{bmatrix}6 & 2\\ 5 & -8\end{bmatrix}$$

حاصل جمعهما سيكون كالتالي:

خطأ!

$$A+B=\begin{bmatrix}1 & 5\\ -2 & 3\end{bmatrix}+\begin{bmatrix}6 & 2\\ 5 & -8\end{bmatrix}=\begin{bmatrix}1+6 & 5+2\\ -2+5 & 3+(-8)\end{bmatrix}=\begin{bmatrix}7 & 7\\ 3 & -5\end{bmatrix}$$

حاصل طرحهما سيكون كالتالي:

خطأ!

$$A-B=A+(-B)=\begin{bmatrix}1 & 5\\ -2 & 3\end{bmatrix}-\begin{bmatrix}6 & 2\\ 5 & -8\end{bmatrix}=\begin{bmatrix}1 & 5\\ -2 & 3\end{bmatrix}+\begin{bmatrix}-6 & -2\\ -5 & 8\end{bmatrix}=\begin{bmatrix}1-6 & 5-2\\ -2-5 & 3+8\end{bmatrix}=\begin{bmatrix}-5 & 3\\ -7 & 11\end{bmatrix}$$



حاصل ضربهما سيكون كالتالي:

$$A = \begin{bmatrix} 1 & 5 \\ -2 & 3 \end{bmatrix} \qquad B = \begin{bmatrix} 6 & 2 \\ 5 & -8 \end{bmatrix}$$

ضرب

$$\begin{pmatrix} (1\ \ 5).(6\ \ 5) & (1\ \ 5).(2\ \ -8) \\ (-2\ \ 3).(6\ \ 5) & (-2\ \ 3).(2\ \ -8) \end{pmatrix}$$

$$\begin{pmatrix} (6 + 25) & (2 + -40) \\ (-12 + 15) & (-4 + -24) \end{pmatrix}$$

$$\begin{pmatrix} 31 & -38 \\ 3 & -28 \end{pmatrix}$$

لنأخذ عملي على ذلك......

لدينا شعاع يحمل الإحداثيات التالية الـ X = ٥ الـ Y= ٠ الـ Z = ٠ الـ W =لنقوم ١
أولاً: بعمل Scaling له بمقدار ١ على ٥ أي ٥/١ على المحاور الثلاثة x,y,z
ثانياً: بعمل Rotation له بمقدار باي / ٤ راديان حول المحور y
ثالثاً: بعمل Translation له بمقدار ١ على المحور x و ٢ على المحور y و -٣ على المحورz
إحسب الإحداثيات الجديدة لهذا الشعاع (الجسم؟)

تكون هذه العملية بشكل تراكمي أي أن أول خطوة نحسب الـ Scalingونأخذ النتيجة ومن ثم نطبق عليها الـ Rotation ونأخذ النتيجة لنطبق عليها الـTranslation
الحل:



Scaling Matrix

$$\mathbf{S}(\mathbf{q})=\begin{bmatrix} q_x & 0 & 0 & 0 \\ 0 & q_y & 0 & 0 \\ 0 & 0 & q_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$S\,(1/5,\ 1/5,\ 1/5)\ \begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$S\,(1/5,\ 1/5,\ 1/5)\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \boxed{= (1, 0, 0, 1)}$$

Rotation Matrix

$$\mathbf{Y}(\vartheta)=\begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boxed{\frac{180}{4} = 45}$$

$$R\,(1, 0, 0, 1)\ \begin{pmatrix} .707 & 0 & -.707 & 0 \\ 0 & 1 & 0 & 0 \\ .707 & 0 & .707 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R\,(1, 0, 0, 1)\ \begin{pmatrix} .707 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -.707 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \boxed{= (.707, 0, -.707, 1)}$$

Translation Matrix

$$\mathbf{T}(\mathbf{p})=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p_x & p_y & p_z & 1 \end{bmatrix}$$

$$T\,(.707,\ 0,\ -.707,\ 1)\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 2 & -3 & 1 \end{pmatrix}$$

$$T\,(.707,\ 0,\ -.707,\ 1)\ \begin{pmatrix} .707 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & -.707 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \boxed{= (.707, 2, -3.707, 1)}$$

إذن إحداثيات الجسم الجديدة بعد عملية الـ **Translation, Rotation, Scaling**

(.707, 2, -3.707, 1)

أو هناك طريقة أخرى للحل وهي ضرب الثلاثة مصفوفات ببعضاها البعض ...

لنرى الكود البرمجي لضرب الثلاثة مصفوفات ببعضها البعض:

كود:

```
device.Transform.World = Matrix.Scaling (2,2,0) * Matrix.RotationX
(Geometry.DegreeToRadian (180))* Matrix.Translation (0,10,1);
```

الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace DirectX_Tutorial

{

public class WinForm : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container components = null;
private float angle ;
public WinForm()
{

InitializeComponent();

this.SetStyle(ControlStyles.AllPaintingInWmPaint |
ControlStyles.Opaque, true);

}


public void InitializeDevice()
```

```
{
PresentParameters presentParams = new
PresentParameters();
presentParams.Windowed = true;
presentParams.SwapEffect = SwapEffect.Discard;
device = new Device(0, DeviceType.Hardware, this,
CreateFlags.SoftwareVertexProcessing, presentParams);
}

protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)
{
device.Transform.Projection =
Matrix.PerspectiveFovLH((float)Math.PI/4, this.Width/this.Height, -1f, 1f);
device.Transform.View = Matrix.LookAtLH(new Vector3(0,0,-
30), new Vector3(0,0,0), new Vector3(0,1,0));
device.Transform.World =
Matrix.Scaling (2,2,0) * Matrix.RotationX   (Geometry.DegreeToRadian
(180))* Matrix.Translation (0,10,1);

device.RenderState.Lighting = false;
device.RenderState.CullMode = Cull.None ;
device.RenderState.Clipping = true;

CustomVertex.PositionColored[] vertices = new
CustomVertex.PositionColored[3];
vertices[0].SetPosition(new Vector3(0f, 0f, 0f));
vertices[0].Color = Color.Red.ToArgb();
```

```
vertices[1].SetPosition(new Vector3(10f, 0f, 0f));

vertices[1].Color = Color.Green.ToArgb();

vertices[2].SetPosition(new Vector3(5f, 10f, 0f));

vertices[2].Color = Color.Yellow.ToArgb();


device.Clear(ClearFlags.Target, Color.DarkSlateBlue , 1.0f, 0);


device.BeginScene();

device.VertexFormat =
CustomVertex.PositionColored.Format;

device.DrawUserPrimitives(PrimitiveType.TriangleList, 1,
vertices);

device.EndScene();



device.Present();


this.Invalidate();
angle += 0.05f;

}


protected override void Dispose (bool disposing)

{

if (disposing)

{

if (components != null)

{

components.Dispose();

}

}
```

```
base.Dispose(disposing);
}

private void InitializeComponent()
{
this.components = new System.ComponentModel.Container();
this.Size = new System.Drawing.Size(500,500);
this.Text = "DirectX Tutorial";
}

static void Main()
{
using (WinForm our_directx_form = new WinForm())
{
our_directx_form.InitializeDevice();
Application.Run(our_directx_form);
}
}
}
}
```

عند عمل RUN للكود بالأعلا فسيظهر لدينا الشكل التالي:



مع تحيات المبرمج هاني العزاوي انتظرونا في الدرس القادم

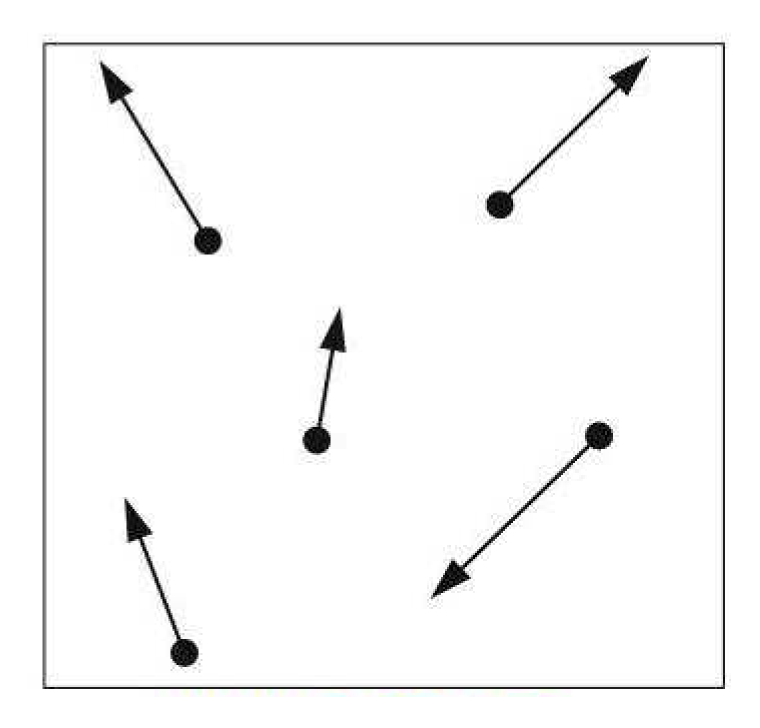


الدرس الحادي عشر:

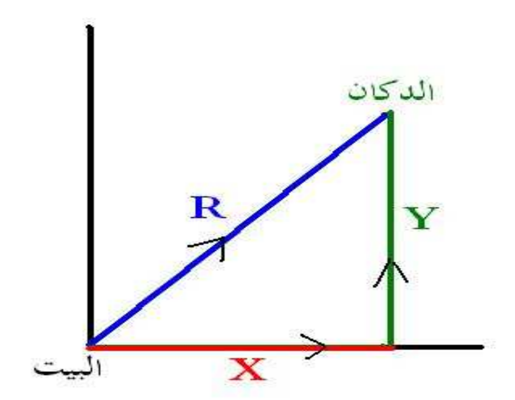
-الـVertex Math
-الـVertex Buffer

-الـVector Math (Vertex)

الـ Vector, وهي عبارة عن نقطة لها إتجاه ومقدار (أنظر إلى الشكل بالأسفل)



لنفرق بين الكميات العددية والكميات المتجهه, أنظر إلى الشكل بالأسفل:



أولاً: القيم العددية:
لنأخذ المثال التالي:

$$\mathbf{u} = (1, 2, 3) \text{ and } \mathbf{v} = (1, 1)$$

$$\|\mathbf{u}\| = \sqrt{u_x^{\ 2} + u_y^{\ 2} + u_z^{\ 2}}$$

$$\|\mathbf{u}\| = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{1 + 4 + 9} = \boxed{\sqrt{14}}$$

$$\|\mathbf{v}\| = \sqrt{1^2 + 1^2} = \boxed{\sqrt{2}}$$

من المثال بالأعلا u و v لنطبق عليها نظرية الـ Normalization Vector وهي أن نجعل جميع أطوال الـ Vectors مساوية لي واحد, وتسمي بي شعاع الواحدة:

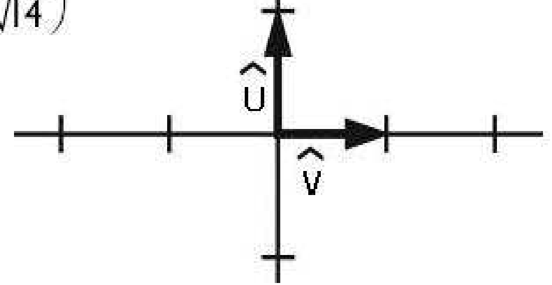
نحسب الـ Normalization في المعادلة التالية

$$\hat{\mathbf{u}} = \frac{\mathbf{u}}{\|\mathbf{u}\|} = \left(\frac{u_x}{\|\mathbf{u}\|}, \ \frac{u_y}{\|\mathbf{u}\|}, \ \frac{u_z}{\|\mathbf{u}\|} \right)$$

حيث أن $\|\mathbf{u}\| = \sqrt{14}, \ \|\mathbf{v}\| = \sqrt{2}$

$$\hat{\mathbf{u}} = \frac{\mathbf{u}}{\sqrt{14}} = \left(\frac{1}{\sqrt{14}}, \ \frac{2}{\sqrt{14}}, \ \frac{3}{\sqrt{14}} \right)$$

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{\sqrt{2}} = \left(\frac{1}{\sqrt{2}}, \ \frac{1}{\sqrt{2}} \right)$$



ثانياً القيم المتجهه:

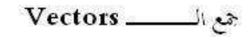
نقول بأن هاذان المتجهان (Vectors) متساويان في عندما تكون:

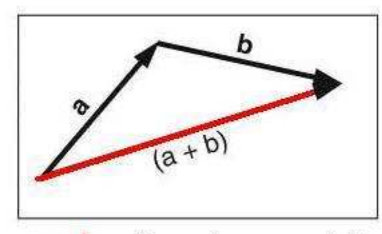
$$(u_x, u_y, u_z) = (v_x, v_y, v_z)$$

تتساوى الـــــــــ v و u في حالة

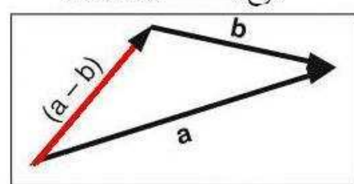
$$u_x = v_x$$
$$u_y = v_y$$
$$u_z = v_z$$

يكون جمع وطرح وضرب الـ Vectors كالتالي:

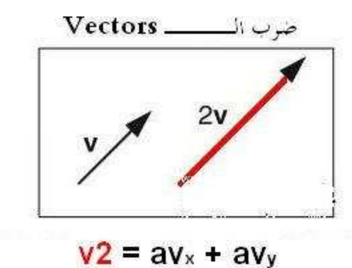




$a + b = (a_x + b_x, a_y + b_y)$



$a - b = a + (-b) = (a_x - b_x, a_y - b_y)$



$v2 = av_x + av_y$

نأتي الآن إلى طريقة الحساب الجبرية, أي إذا كان هناك زاوية بين

المتجهيين...

قانون جمع المتجهات في حالة وجود زاوية تحصرها

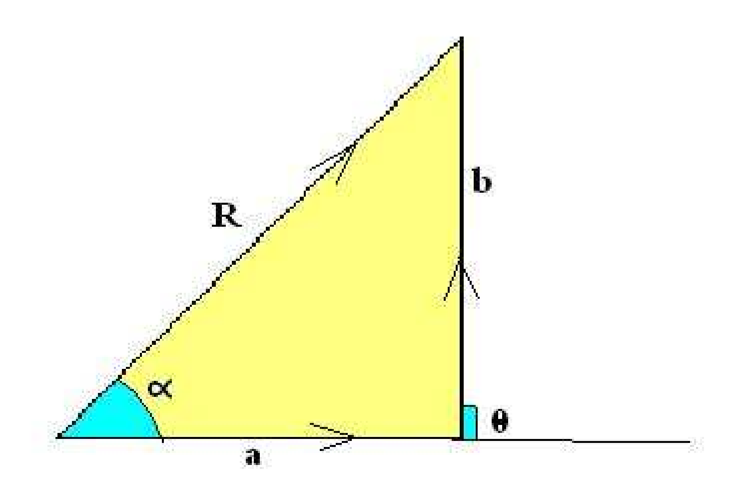
$$R= (a)^2 + (b)^2 + 2 (a)(b) \cos\theta$$

قانون طرح المتجهات في حالة وجود زاوية تحصرها

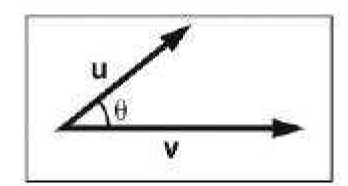
$$R = (a)^2 + (b)^2 - 2 (a)(b) \cos\theta$$

قانون حساب الزاوية الداخلية

$$\sin\alpha = \frac{b}{R}$$



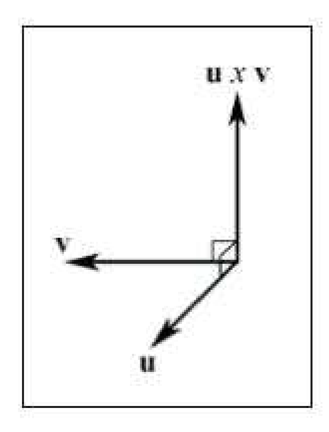




$$\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\| \|\mathbf{v}\| \cos \vartheta$$

النتيجة ستكون قيمة عددية أو مقدارية

Cross Products



$$p_x = (u_y v_z - u_z v_y)$$

$$p_y = (u_z v_x - u_x v_z)$$

$$p_z = (u_x v_y - u_y v_x)$$

النتيجة ستكون قيمة متجهه

-الـVertex Buffer

وهي فئة (Class) موجوده في الـ Direct٣d تعمل على حجز منطقة في ذاكرة الـ RAM أو في كرت الشاشة, بشكل مؤقت من أجل عملية تخزين الـ Vertex لتسهيل التعامل وإجراء العمليات عليها.

كود:

```
new VertexBuffer(typeof(CustomVertex.PositionColored ), 3, device,
Usage.Dynamic |
```

```
Usage.WriteOnly, CustomVertex.PositionColored.Format, Pool.Default);
```

كما نرى... تحمل هذه الفئة عدة باراميترات:

الأول: لتحديد نوع الـ Vertex المراد التعامل معه (PositionColored, PositionNormal, ... etc) ,خذها قاعدة إذا قمت بتحديد نوع الـ Vertex وليكن PositionColored فهذا يعني بأنك ستصرح بجميع أجزاء الكود عن هذا الـ Vectex نفسه.

الثاني: وهو عدد الـ Vertex وعادةً نضع فيه الـMaximum Number

الثالث: الـ Device وهو تابع للكائن الذي أنشأناه عندما تكلمنا عن كرت الشاشة.

الرابع: الـ Usage وهو الذي يحدد الصلاحيات, ومنها..
DoNotClipوهو الخيار الذي لا يسمح للـ Vertex بي خاصية الـClipping
Dynamicوهو الخيار الذي يتيح بحجز منطقة ثابته الحجم والعنوان في الذاكرة.
Pointsوهو الخيار الذي يستخدم عند التعامل مع النقاط فقط.
SoftwareProcessingوهو الخيار الذي تتم فيه معالجة الـ Vertex في الـ Software
WriteOnlyوهو الخيار الذي يستخدم من أجل منع القراءة من الـ Buffer

الخامس: لتحديد نوع الـ Vertex المراد تخزينة بداخل الـ Buffer

السادس Pool: لتحديد مكان الـ Buffer وهناك عدة خيارات:
Default : يعتمد هذا الخيار على الـ Usage والذي حددناه من قبل.
Managed : تكون عملية تخزين الـ Vertex بداخل الـVGA
SystemMemory : تكون عملية تخزين الـ Vertex بداخل الـRAM

تأتي بعدها عملية الـ Lock والتي تعني بالسماح للـ CPU بالدخول إلى العتاد VGAمن أجل تخزين الـ Vertex, وعند إنتهاء هذه العملية نقوم بعمل Unlock

كود:

```
CustomVertex.PositionColored  [] verts =
(CustomVertex.PositionColored [])vb.Lock  (0,0);
            verts[0].SetPosition(new Vector3(0.0f, 1.0f,
            1.0f));
                    verts[0].Color  =
System.Drawing.Color.Aqua.ToArgb();
            verts[1].SetPosition(new Vector3(-1.0f, -1.0f,
            1.0f));
                    verts[1].Color =
System.Drawing.Color.Black.ToArgb();
            verts[2].SetPosition(new Vector3(1.0f, -1.0f,
            1.0f));
                    verts[2].Color =
System.Drawing.Color.Purple.ToArgb();
```

```
vb.Unlock ();
```

ليصبح لدي الكود كاملاً كالتالي :

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container
components = null;
private float angle;
private VertexBuffer vb;




public Form1()
{
InitializeComponent();
}
public void ondeivce()
{
PresentParameters pp = new PresentParameters
();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat = DepthFormat.D16
;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);
}
public void camera()
```

```
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,3),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.RenderState.Lighting = false;
device.RenderState.CullMode = Cull.None;
}
public void position()
{

vb = new
VertexBuffer(typeof(CustomVertex.PositionColored  ), 3, device,
Usage.SoftwareProcessing    |
Usage.WriteOnly,
CustomVertex.PositionColored.Format, Pool.SystemMemory );

CustomVertex.PositionColored  [] verts =
(CustomVertex.PositionColored [])vb.Lock  (0,0);
verts[0].SetPosition(new Vector3(0.0f, 1.0f,
1.0f));
verts[0].Color  =
System.Drawing.Color.Aqua.ToArgb();
verts[1].SetPosition(new Vector3(-1.0f, -1.0f,
1.0f));
verts[1].Color =
System.Drawing.Color.Black.ToArgb();
verts[2].SetPosition(new Vector3(1.0f, -1.0f,
1.0f));
verts[2].Color =
System.Drawing.Color.Purple.ToArgb();

vb.Unlock ();



}
public void render()
{
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.Black ,1,1);
device.BeginScene ();

device.SetStreamSource(0, vb, 0);

device.VertexFormat =
CustomVertex.PositionColored .Format ;

device.DrawPrimitives(PrimitiveType.TriangleList, 0, 1);
```

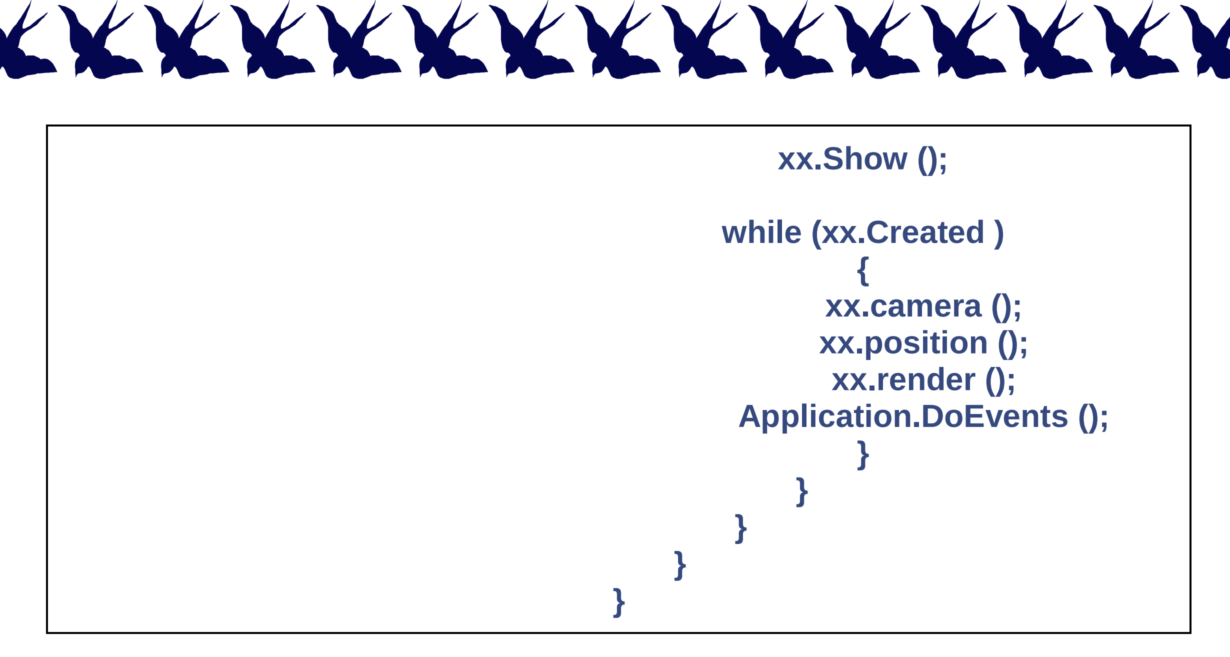
```
device.Transform.World =  Matrix.RotationAxis
(new Vector3 (1,2,3),angle);

device.EndScene ();
device.Present ();
angle += 0.05f;
}
protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}


protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
this.components = new
System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion

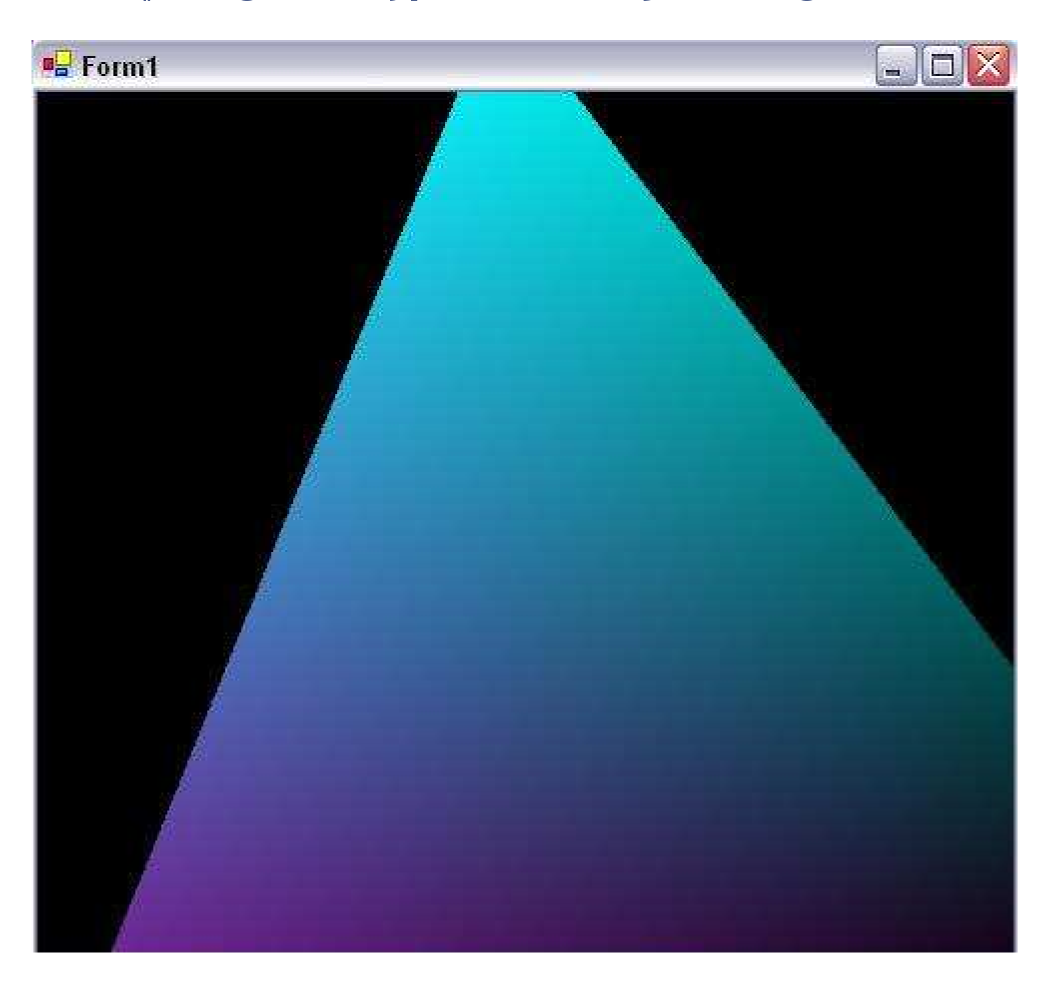

static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondeivce ();
```



```
xx.Show ();

while (xx.Created )
{
xx.camera ();
xx.position ();
xx.render ();
Application.DoEvents ();
}
}
}
}
}
```

عند عمل RUN للكود بالأعلا فسيظهر لنا الشكل التالي:



نلاحظ في هذا الكود بعض التغيرات...
حيث قمنا بعمل Function خاصة للـ Camera وأخرى خاصة بي الـ Vertex
وأخرى بي الـ ... Render وبذلك خففنا الضغط على الدالةOnPaint

وقمنا ايضاً بي إستبدال هذه الأجزاء من الكود والخاصة بعملية إستمرارية العرض:

كود:

```
this.SetStyle(ControlStyles.AllPaintingInWmPaint |
ControlStyles.Opaque, true);
```

كود:

```
Application.Run(our_directx_form);
```

كود:

```
this.Invalidate();
```

بي التالي, والتي تعطي نتائج أفضل:

كود:

```
Show ();
while (xx.Created )
{
...
...
Application.DoEvents ();
}
```

مثال آخر لعمل مكعب بواسطة ١٢ مثلث, وتوضع الإحداثيات يكون على دائرة الوحدة.

الكود:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container
components = null;
private float angle;
private VertexBuffer vb;
```

```
public Form1()
{
InitializeComponent();
}
public void ondeivce()
{
PresentParameters pp = new PresentParameters
();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat = DepthFormat.D16
;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);
}
public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,6),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.Transform.World =  Matrix.RotationAxis
(new Vector3 (1,2,3),angle);
device.RenderState.Lighting = false;
device.RenderState.CullMode = Cull.None;
}
public void position()
{

vb = new
VertexBuffer(typeof(CustomVertex.PositionColored  ), 36, device,
Usage.SoftwareProcessing    |
Usage.WriteOnly,
CustomVertex.PositionColored.Format, Pool.SystemMemory );

CustomVertex.PositionColored  [] verts =
(CustomVertex.PositionColored [])vb.Lock  (0,0);

//Front face
//المثلث الأول
verts[0] = new CustomVertex.PositionColored(-
1.0f, 1.0f, 1.0f, Color.Red.ToArgb());
verts[1] = new CustomVertex.PositionColored(-
1.0f, -1.0f, 1.0f, Color.Red.ToArgb());
verts[2] = new
CustomVertex.PositionColored(1.0f, 1.0f, 1.0f, Color.Red.ToArgb());
//المثلث الثاني
verts[3] = new CustomVertex.PositionColored(-
1.0f, -1.0f, 1.0f, Color.Red.ToArgb());
```

```
verts[4] = new CustomVertex.PositionColored(1.0f, -1.0f, 1.0f, Color.Red.ToArgb());
verts[5] = new CustomVertex.PositionColored(1.0f, 1.0f, 1.0f, Color.Red.ToArgb());

// Back face
//المثلث الأول//
verts[6] = new CustomVertex.PositionColored(-1.0f, 1.0f, -1.0f, Color.Blue.ToArgb());
verts[7] = new CustomVertex.PositionColored(1.0f, 1.0f, -1.0f, Color.Blue.ToArgb());
verts[8] = new CustomVertex.PositionColored(-1.0f, -1.0f, -1.0f, Color.Blue.ToArgb());
//المثلث الثاني//
verts[9] = new CustomVertex.PositionColored(-1.0f, -1.0f, -1.0f, Color.Blue.ToArgb());
verts[10] = new CustomVertex.PositionColored(1.0f, 1.0f, -1.0f, Color.Blue.ToArgb());
verts[11] = new CustomVertex.PositionColored(1.0f, -1.0f, -1.0f, Color.Blue.ToArgb());

// Top face
//المثلث الأول//
verts[12] = new CustomVertex.PositionColored(-1.0f, 1.0f, 1.0f, Color.Yellow.ToArgb());
verts[13] = new CustomVertex.PositionColored(1.0f, 1.0f, -1.0f, Color.Yellow.ToArgb());
verts[14] = new CustomVertex.PositionColored(-1.0f, 1.0f, -1.0f, Color.Yellow.ToArgb());
//المثلث الثاني//
verts[15] = new CustomVertex.PositionColored(-1.0f, 1.0f, 1.0f, Color.Yellow.ToArgb());
verts[16] = new CustomVertex.PositionColored(1.0f, 1.0f, 1.0f, Color.Yellow.ToArgb());
verts[17] = new CustomVertex.PositionColored(1.0f, 1.0f, -1.0f, Color.Yellow.ToArgb());

// Bottom face
//المثلث الأول//
verts[18] = new CustomVertex.PositionColored(-1.0f, -1.0f, 1.0f, Color.Black.ToArgb());
verts[19] = new CustomVertex.PositionColored(-1.0f, -1.0f, -1.0f, Color.Black.ToArgb());
verts[20] = new CustomVertex.PositionColored(1.0f, -1.0f, -1.0f, Color.Black.ToArgb());
//المثلث الثاني//
verts[21] = new CustomVertex.PositionColored(-1.0f, -1.0f, 1.0f, Color.Black.ToArgb());
verts[22] = new CustomVertex.PositionColored(1.0f, -1.0f, -1.0f, Color.Black.ToArgb());
```

```
verts[23] = new
CustomVertex.PositionColored(1.0f, -1.0f, 1.0f, Color.Black.ToArgb());

// **** face
//المثلث الأول
verts[24] = new CustomVertex.PositionColored(-
1.0f, 1.0f, 1.0f, Color.Gray.ToArgb());
verts[25] = new CustomVertex.PositionColored(-
1.0f, -1.0f, -1.0f, Color.Gray.ToArgb());
verts[26] = new CustomVertex.PositionColored(-
1.0f, -1.0f, 1.0f, Color.Gray.ToArgb());
//المثلث الثاني
verts[27] = new CustomVertex.PositionColored(-
1.0f, 1.0f, -1.0f, Color.Gray.ToArgb());
verts[28] = new CustomVertex.PositionColored(-
1.0f, -1.0f, -1.0f, Color.Gray.ToArgb());
verts[29] = new CustomVertex.PositionColored(-
1.0f, 1.0f, 1.0f, Color.Gray.ToArgb());

// Right face
//المثلث الأول
verts[30] = new
CustomVertex.PositionColored(1.0f, 1.0f, 1.0f, Color.Green.ToArgb());
verts[31] = new
CustomVertex.PositionColored(1.0f, -1.0f, 1.0f, Color.Green.ToArgb());
verts[32] = new
CustomVertex.PositionColored(1.0f, -1.0f, -1.0f, Color.Green.ToArgb());
//المثلث الثاني
verts[33] = new
CustomVertex.PositionColored(1.0f, 1.0f, -1.0f, Color.Green.ToArgb());
verts[34] = new
CustomVertex.PositionColored(1.0f, 1.0f, 1.0f, Color.Green.ToArgb());
verts[35] = new
CustomVertex.PositionColored(1.0f, -1.0f, -1.0f, Color.Green.ToArgb());


vb.Unlock ();



}
public void render()
{
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.Black  ,1,1);
device.BeginScene ();

device.SetStreamSource(0, vb, 0);
```

```
device.VertexFormat =
CustomVertex.PositionColored .Format ;

device.DrawPrimitives(PrimitiveType.TriangleList, 0, 12);


device.EndScene ();
device.Present ();
angle += 0.05f;
}
protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}


protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
this.components = new
System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion


static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondeivce ();
```

```
xx.Show ();

while (xx.Created )
{
xx.camera ();
xx.position ();
xx.render ();
Application.DoEvents ();
}
}
}
}
}
```

عند عمل RUN سيظهر الشكل التالي:

Form1

WWW.JASPH.NET

Form1

ملاحظة: في الكود بالأعلا قمت بتغير طريقة التصريح عن إحداثيات الـ Vertexفبدلاً من كتابته هكذا...

كود:

```
verts[0].SetPosition (new Vector3 (-1.0f, 1.0f, 1.0f));
verts[0].Color = Color.Red.ToArgb
();
```

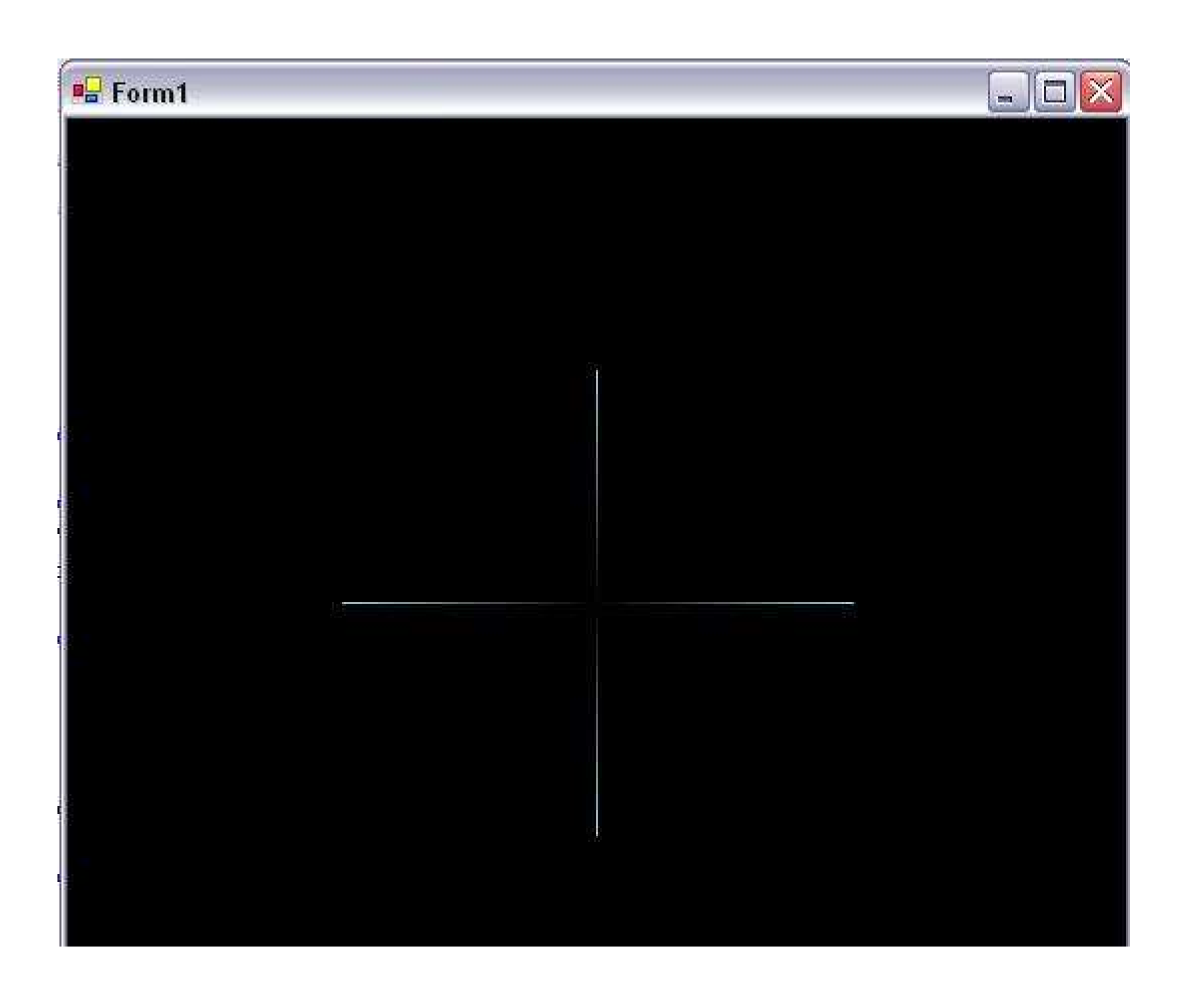
قمت بكتابته بهذه الطريقة..

كود:

```
verts[0] = new CustomVertex.PositionColored(-1.0f, 1.0f, 1.0f,
Color.Red.ToArgb());
```

فكل الطرق تأدي إلى روما ...... ولأكن إذا سألتني أيهم أفضل بالنسبة لعملية الصيانه والذاكرة .... سأقول لك الطريقة الثانية

لنأخذ مثال آخر...
لنقل أني أريد رسم أربعة خطوط متقاطعة كما في الشكل بالأسفل:





لعمل ذالك:

أولاً: سنقوم بعمل VertexBuffer, ونحدد حجمة بي ٩ أي أكثر من المطلوب بواحد حيث نلاحظ بأن عدد النقاط في الشكل بالأعلا هي ٤ ولكل خط نقطتان إذا يصبح ٨

كود:

```
vb = new VertexBuffer (typeof(CustomVertex.PositionNormalColored
),100,device,Usage.WriteOnly
,CustomVertex.PositionNormalColored.Format ,Pool.Default );
CustomVertex.PositionNormalColored [] cc =
(CustomVertex.PositionNormalColored [])vb.Lock (0,0);
```

ثانياً :لتوفير الوقت والجهد سنستخدم من الجمل التكرارية جملة for وبما أني أريد رسم أربع نقاط فستصبح كالتالي:

كود:

```
for (int i =0;i<=4;i++)
```

ثالثاً :إيجاد الزاوية عن طريق
(PI*٢) ٣٦٠ضرب قيمة العدد (i) تقسيم مجموع النقاط المراد رسمها وهي ٤

كود:

```
float theta = (float)(2 * Math.PI * i) / 4;
```

رابعاً :إيجاد الإحداثيات النقطة x بي cos الزاوية والنقطة y بي sin الزاوية.

كود:

```
cc[i*2].SetPosition (new Vector3 ((float)Math.Cos  (theta),(float)Math.Sin
(theta),0));
```

خامساً :نقوم بتحديد عدد النقاط المراد رسمها في الدالة DrawPrimitives ويكون عدد النقاط المراد التوصيل بينها وهو ٤ ضرب ٢ وتساوي ٨

كود:

```
device.DrawPrimitives (PrimitiveType.LineStrip   ,0,8);
```

الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
```

```
using Microsoft.DirectX.Direct3D;


namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container
components = null;
private float angle;
private VertexBuffer vb;

public Form1()
{
InitializeComponent();
}
public void ondeivce()
{
PresentParameters pp = new PresentParameters
();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat = DepthFormat.D16
;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);
}
public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,5),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.RenderState.Lighting = false;
device.RenderState.CullMode = Cull.None;
}
public void position()
{
vb = new VertexBuffer
(typeof(CustomVertex.PositionNormalColored
),9,device,Usage.WriteOnly
,CustomVertex.PositionNormalColored.Format ,Pool.Default );
CustomVertex.PositionNormalColored [] cc =
(CustomVertex.PositionNormalColored [])vb.Lock (0,0);
for (int i =0;i<=4;i++)
{
float theta = (float)(2 * Math.PI * i) / 4;
```

```
cc[i*2].SetPosition (new Vector3
((float)Math.Cos  (theta),(float)Math.Sin  (theta),0));
cc[i*2].Color = Color.SkyBlue .ToArgb ();

}
vb.Unlock ();
}
public void render()
{
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.Black   ,1,1);
device.BeginScene ();

device.SetStreamSource (0,vb,0);
device.VertexFormat =
CustomVertex.PositionNormalColored.Format ;
device.DrawPrimitives (PrimitiveType.LineStrip
,0,8);
device.Transform.World =  Matrix.RotationAxis
(new Vector3 (0,0,3),angle);


device.EndScene ();
device.Present ();
angle += 0.05f;
}
protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}


protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

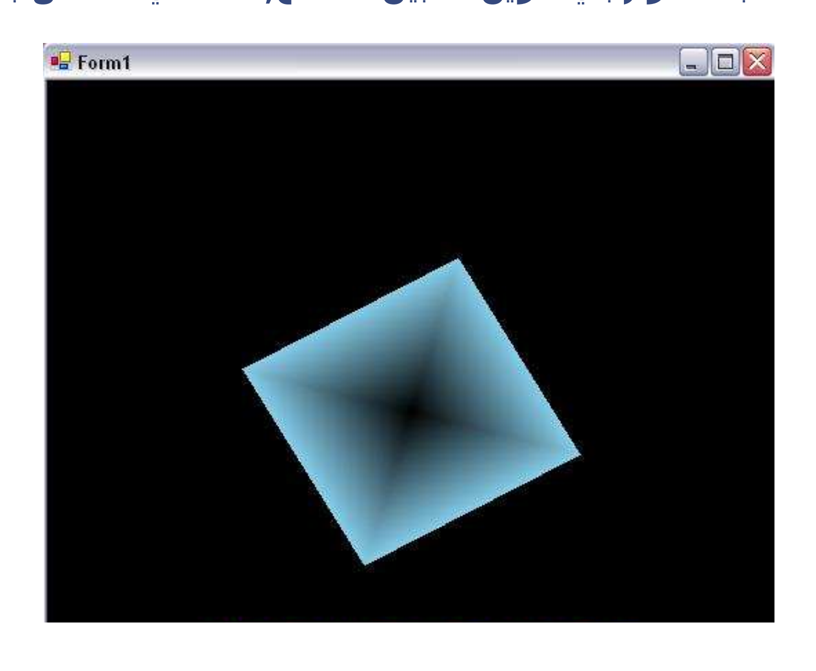
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
```

```
/// </summary>
private void InitializeComponent()
{
this.components = new
System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion


static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondeivce ();
xx.Show ();

while (xx.Created )
{
xx.camera ();
xx.position ();
xx.render ();
Application.DoEvents ();
}
}
}
}
}
```

ما رأيك الآن بأن نقوم بي تلوين ما بين الأضلع, كما في الشكل بالأسفل:



سنقوم فقط بتغير الـ DrawPrimitives من

كود:

```
device.DrawPrimitives (PrimitiveType.LineStrip ,0,8);
```

إلى:

كود:

```
device.DrawPrimitives (PrimitiveType.TriangleStrip   ,0,8);
```

وأيضاً سيكون حجم الـ VertexBuffer أكبر بمرتين وذلك لأنه سيأخذ الألوان والإحداثيات أي ٩ ضرب ٢ وتساوي ١٨

الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container components = null;
private float angle;
private VertexBuffer vb;

public Form1()
{
InitializeComponent();
}
public void ondeivce()
{
PresentParameters pp = new PresentParameters ();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
pp.EnableAutoDepthStencil = true;
```

```
pp.AutoDepthStencilFormat = DepthFormat.D16
;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);
}
public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,5),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.RenderState.Lighting = false;
device.RenderState.CullMode = Cull.None;
}
public void position()
{
vb = new VertexBuffer
(typeof(CustomVertex.PositionNormalColored
),18,device,Usage.WriteOnly
,CustomVertex.PositionNormalColored.Format ,Pool.Default );
CustomVertex.PositionNormalColored [] cc =
(CustomVertex.PositionNormalColored [])vb.Lock (0,0);
for (int i =0;i<=4;i++)
{
float theta = (float)(2 * Math.PI * i) / 4;
cc[i*2].SetPosition (new Vector3
((float)Math.Cos  (theta),(float)Math.Sin  (theta),0));
cc[i*2].Color = Color.SkyBlue .ToArgb ();

}
vb.Unlock ();
}
public void render()
{
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.Black   ,1,1);
device.BeginScene ();

device.SetStreamSource (0,vb,0);
device.VertexFormat =
CustomVertex.PositionNormalColored.Format ;
device.DrawPrimitives
(PrimitiveType.TriangleStrip   ,0,8);
device.Transform.World =  Matrix.RotationAxis
(new Vector3 (0,0,3),angle);


device.EndScene ();
device.Present ();
angle += 0.05f;
```

```
}
protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}


protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
this.components = new
System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion


static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondeivce ();
xx.Show ();

while (xx.Created )
{
xx.camera ();
xx.position ();
xx.render ();
Application.DoEvents ();
}
```

```
}
}
}
}
```

مثال آخر: لنقم برسم دائرة:

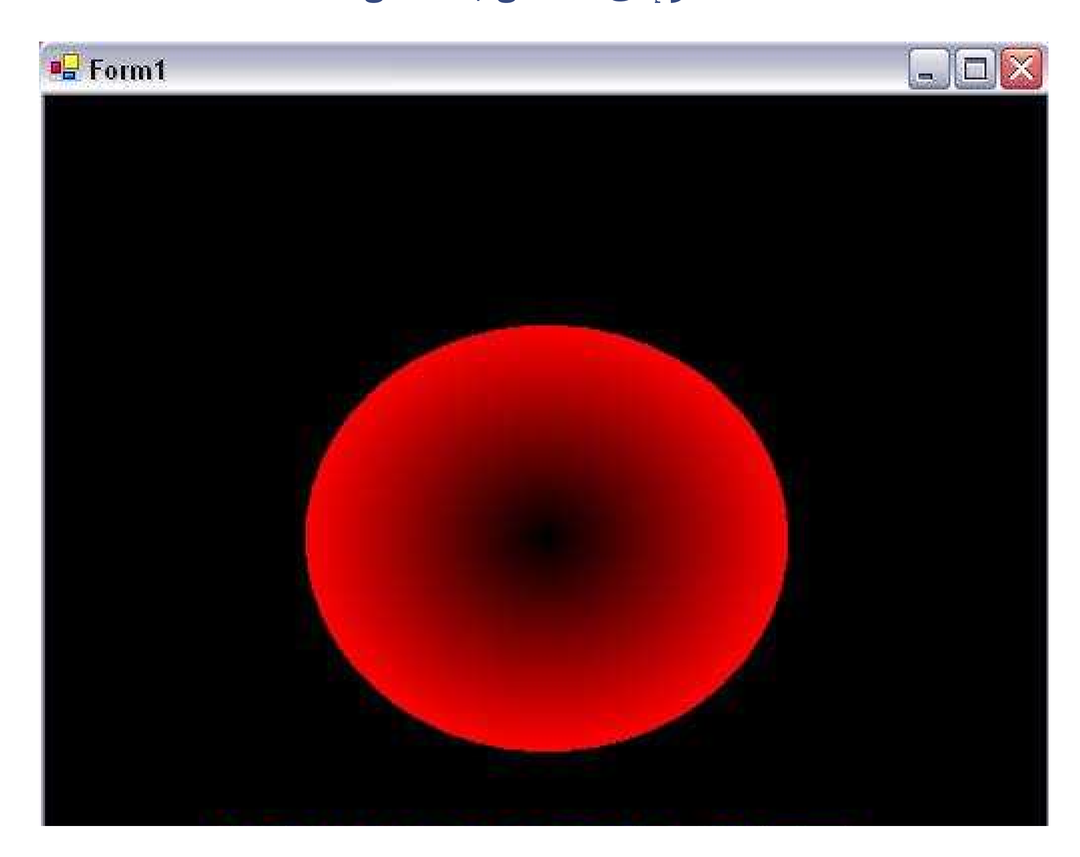
بنفس النظرية بالأعلا ولأكن سنقوم برسم خطوط أكثر وذلك بتكبير

الـ VertexBuffer ليصبح ١٢٠

والـ for لتصبح٥٠

والـ DrawPrimitives ليصبح ١٠٠

أنظر إلى الشكل بالأسفل:



الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;
```

```
namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container
components = null;
private float angle;
private VertexBuffer vb;

public Form1()
{
InitializeComponent();
}
public void ondeivce()
{
PresentParameters pp = new PresentParameters
();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat = DepthFormat.D16
;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);
}
public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,5),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.RenderState.Lighting = false;
device.RenderState.CullMode = Cull.None;
}
public void position()
{
vb = new VertexBuffer
(typeof(CustomVertex.PositionNormalColored
),120,device,Usage.WriteOnly
,CustomVertex.PositionNormalColored.Format ,Pool.Default );
CustomVertex.PositionNormalColored [] cc =
(CustomVertex.PositionNormalColored [])vb.Lock (0,0);
for (int i =0;i<=50;i++)
{
float theta = (float)(2 * Math.PI * i) / 50;
cc[i*2].SetPosition (new Vector3
((float)Math.Cos  (theta),(float)Math.Sin  (theta),0));
cc[i*2].Color = Color.Red  .ToArgb ();
```

```
}
vb.Unlock ();
}
public void render()
{
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.Black   ,1,1);
device.BeginScene ();

device.SetStreamSource (0,vb,0);
device.VertexFormat =
CustomVertex.PositionNormalColored.Format ;
device.DrawPrimitives
(PrimitiveType.TriangleStrip   ,0,100);
device.Transform.World =  Matrix.RotationAxis
(new Vector3 (1,1,3),angle);


device.EndScene ();
device.Present ();
angle += 0.05f;
}
protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}


protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
```

```
this.components = new
System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion


static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondeivce ();
xx.Show ();

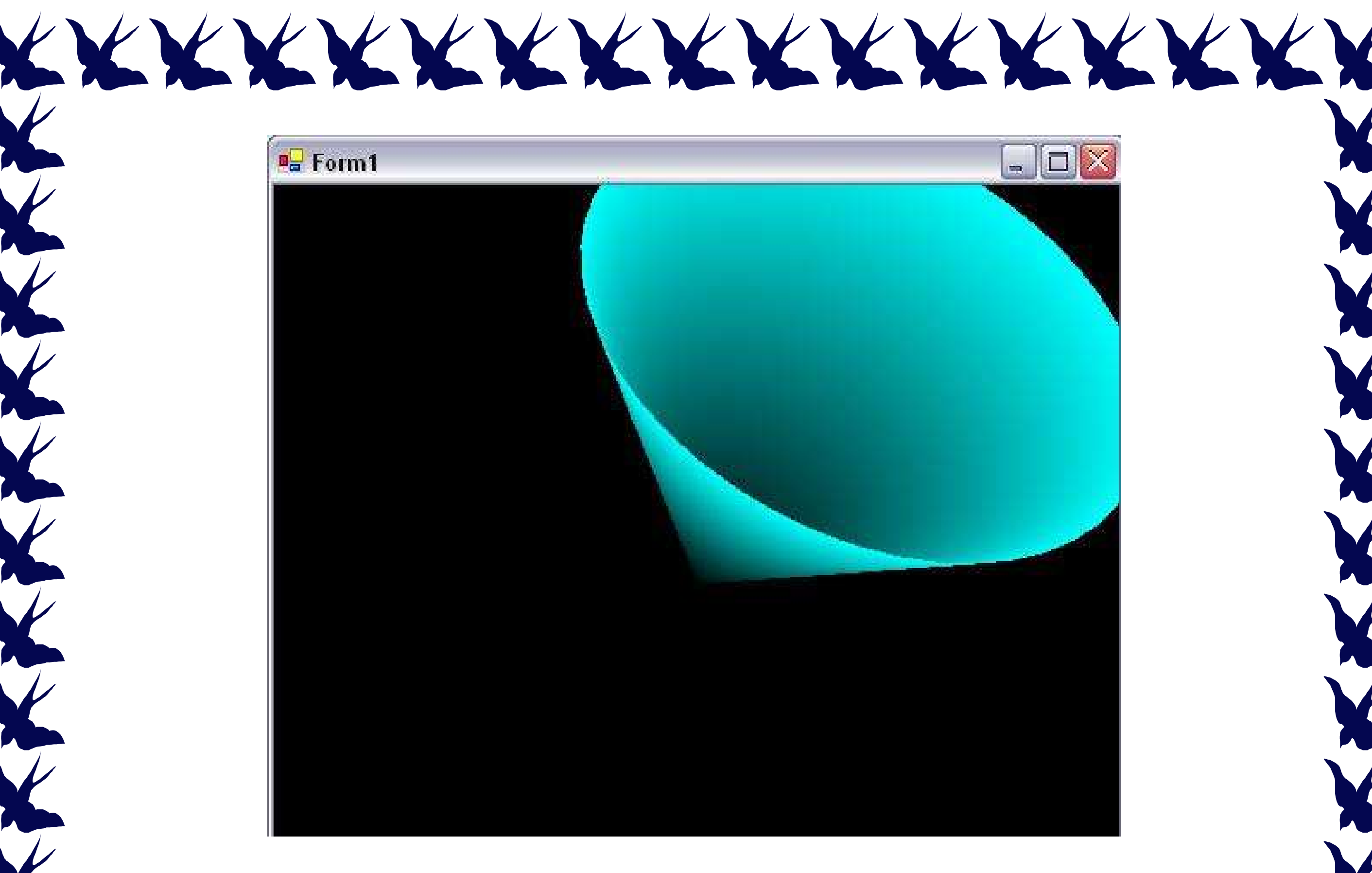
while (xx.Created )
{
xx.camera ();
xx.position ();
xx.render ();
Application.DoEvents ();
}
}
}
}
}
```

مثال: لرسم قمع, وبنفس النظرية:
حيث سنقوم بي إضافة

كود:

```
cc[i*2].SetPosition (new Vector3 ((float)Math.Cos  (theta),(float)Math.Sin
(theta),0));
cc[i*2].SetPosition (new Vector3 ((float)Math.Cos  (theta),(float)Math.Sin
(theta),2));
```

ليظهر لنا الشكل التالي:



الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container
components = null;
private float angle;
private VertexBuffer vb;

public Form1()
{
InitializeComponent();
}
```

```
public void ondeivce()
{
PresentParameters pp = new PresentParameters
();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat = DepthFormat.D16
;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);
}
public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,5),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.RenderState.Lighting = false;
device.RenderState.CullMode = Cull.None;
}
public void position()
{
vb = new VertexBuffer
(typeof(CustomVertex.PositionNormalColored
),120,device,Usage.WriteOnly
,CustomVertex.PositionNormalColored.Format ,Pool.Default );
CustomVertex.PositionNormalColored [] cc =
(CustomVertex.PositionNormalColored [])vb.Lock (0,0);
for (int i =0;i<=50;i++)
{
float theta = (float)(2 * Math.PI * i) / 50;

cc[i*2].SetPosition (new Vector3
((float)Math.Cos  (theta),(float)Math.Sin  (theta),0));
cc[i*2].SetPosition (new Vector3
((float)Math.Cos  (theta),(float)Math.Sin  (theta),2));
cc[i*2].Color = Color.Aqua  .ToArgb ();



}
vb.Unlock ();
}
public void render()
{
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.Black   ,1,1);
device.BeginScene ();
```

```
device.SetStreamSource (0,vb,0);
device.VertexFormat =
CustomVertex.PositionNormalColored.Format ;
device.DrawPrimitives
(PrimitiveType.TriangleStrip   ,0,100);
device.Transform.World =  Matrix.RotationAxis
(new Vector3 (3,3,3),angle);


device.EndScene ();
device.Present ();
angle += 0.05f;
}
protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}


protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

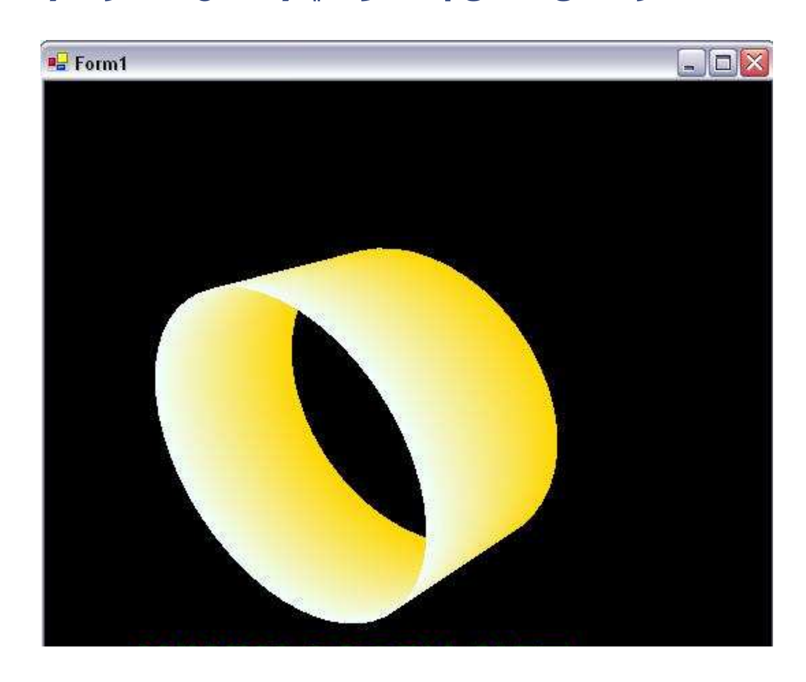
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
this.components = new
System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion


static void Main()
{
```

```
using (Form1 xx = new Form1 ())
{
xx.ondeivce ();
xx.Show ();

while (xx.Created )
{
xx.camera ();
xx.position ();
xx.render ();
Application.DoEvents ();
}
}
}
}
}
```

مثال آخر لعمل شكل إسطواني (بنفس النظرية):



الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
```

```
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container
components = null;
private float angle;
private VertexBuffer vb;

public Form1()
{
InitializeComponent();
}
public void ondeivce()
{
PresentParameters pp = new PresentParameters
();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat = DepthFormat.D16
;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);
}
public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,5),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.RenderState.Lighting = false;
device.RenderState.CullMode = Cull.None;
}
public void position()
{
vb = new VertexBuffer
(typeof(CustomVertex.PositionNormalColored
),120,device,Usage.WriteOnly
,CustomVertex.PositionNormalColored.Format ,Pool.Default );
CustomVertex.PositionNormalColored [] cc =
(CustomVertex.PositionNormalColored [])vb.Lock (0,0);
```

```
for (int i =0;i<=50;i++)
{
float theta = (float)(2 * Math.PI * i) / 50;

cc[i*2].SetPosition (new Vector3
((float)Math.Cos  (theta),(float)Math.Sin  (theta),0));
cc[i*2].SetPosition   (new Vector3
((float)Math.Cos  (theta),(float)Math.Sin  (theta),0));
cc[i*2].Color = Color.Gold    .ToArgb ();

cc[i*2+1].SetPosition (new Vector3
((float)Math.Cos  (theta),(float)Math.Sin  (theta),1));
cc[i*2+1].SetPosition   (new Vector3
((float)Math.Cos  (theta),(float)Math.Sin  (theta),1));
cc[i*2+1].Color = Color.Azure   .ToArgb
();



}
vb.Unlock ();
}
public void render()
{
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.Black   ,1,1);
device.BeginScene ();

device.SetStreamSource (0,vb,0);
device.VertexFormat =
CustomVertex.PositionNormalColored.Format ;
device.DrawPrimitives
(PrimitiveType.TriangleStrip   ,0,100);
device.Transform.World =  Matrix.RotationAxis
(new Vector3 (3,3,3),angle);


device.EndScene ();
device.Present ();
angle += 0.05f;
}
protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}


protected override void Dispose( bool disposing )
{
```

```
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
this.components = new
System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion
static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondeivce ();
xx.Show ();

while (xx.Created )
{
xx.camera ();
xx.position ();
xx.render ();
Application.DoEvents ();
}
}
}
}
}
```

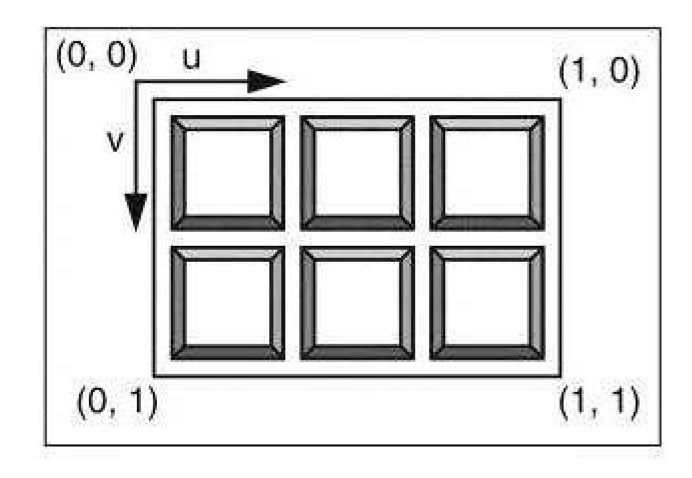


الدرس الثاني عشر:

-الـTexture
-أنواع الـVertex

-الـTexture

تعلمنا سابقاً كيف أنه يمكننا إعطاء الـ Vertex لون بالإعتماد على إحداثياته,
بنفس هذه النظرية ستستخدم الـTexture
الـ Texture نستخدم هذا المصطلح في الـ DirectX عندما ندخل الصور تحمل الإمتداد jpg أو Bitmap إلى الـObjects
تستخدم الإحداثيات الـ V و الـ U من أجل عملية التحكم في إحداثيات الصور .
أنظر إلى الشكل بالأسفل:



أي لو أخذت هذه الصور في الأسفل فستصبح الإحداثيات عليها كالتالي:



الآن لنأخذ التطبيق العملي, من أجل إدخال الصورة على أي كائن.....

أولاً: نقوم بوضع هذه الصورة في الأسفل بداخل نفس ملف المشروع, ونعطيها الإسم RAAD



ثانياً: نقوم بالتصريح عن المتغير Texture كالتالي:

private Texture txt;

ثالثاً: نستخدم الفئة (Class) المسمى TextureLoader مع الدالة (Function)المسمى FromFile من أجل عملية تحديد مسار الصورة المراد التعامل معها ,وتخزينها في الذاكرة .

كود:

```
txt = TextureLoader.FromFile (device, Application.StartupPath +
@"\..\..\RAAD.jpg");
```

حيث أن Application.StartupPath يمثل المسار الحالي للمشروع, أي يجب وضع الصورة بنفس مسار الملف الذي خزنت فيه المشروع, و RAAD.jpg يمثل إسم الصورة مع الإمتداد والتي وضعنها بالخطوة الأولى.

رابعاً: نقوم بتغير نوع الـ Vertex أينما وجد من PositionColored إلى الـ PositionTextured ,فبهذه الخطوة قمنا بربط نقاط الـ Vertex بي الـ Texture.لنرى المثال بالأسفل والذي يمثل نقاط لمثلث وتوضع إحداثيات الصورة (Texture) عليمها بالإعتماد على الـ v و الـu

كود:

```
verts[0].SetPosition(new Vector3(0, 0f, 0));
verts[0].Tv = 0.5f;
verts[0].Tu = 0.5f;

verts[1].SetPosition(new Vector3(-1, -1, 0));
verts[1].Tv = 1;
verts[1].Tu = 0;

verts[2].SetPosition(new Vector3(1, -1.0f,0));
verts[2].Tv = 1;
verts[2].Tu = 1;
```

خامساً: تبقى علينا عملية قراءة الصورة وذلك بإستخدام الكود التالي:

كود:

```
device.SetTexture(0,txt);
```

الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
```

```
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container
components = null;
private float angle;
private VertexBuffer vb;
private Texture txt;



public Form1()
{
InitializeComponent();
}
public void ondeivce()
{
PresentParameters pp = new PresentParameters
();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat = DepthFormat.D16
;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);
txt =  TextureLoader.FromFile  (device,
Application.StartupPath + @"\..\..\RAAD.jpg");
}
public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,3),new Vector3 (0,0,0),new Vector3 (0,1,0));
//device.Transform.World =  Matrix.RotationAxis
(new Vector3 (5,2,3),angle);
device.RenderState.Lighting = false;
device.RenderState.CullMode = Cull.None;
}
```

```
public void position()
{

vb = new
VertexBuffer(typeof(CustomVertex.PositionTextured ), 3, device,
Usage.SoftwareProcessing   |
Usage.WriteOnly,
CustomVertex.PositionTextured.Format, Pool.SystemMemory );

CustomVertex.PositionTextured [] verts =
(CustomVertex.PositionTextured [])vb.Lock (0,0);

verts[0].SetPosition(new Vector3(0, 0f, 0));
verts[0].Tv = 0.5f;
verts[0].Tu = 0.5f;

verts[1].SetPosition(new Vector3(-1, -1, 0));
verts[1].Tv = 1;
verts[1].Tu = 0;

verts[2].SetPosition(new Vector3(1, -1.0f,0));
verts[2].Tv = 1;
verts[2].Tu = 1;


vb.Unlock ();



}
public void render()
{
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.SkyBlue  ,1,1);
device.BeginScene ();

device.SetStreamSource(0, vb, 0);
device.SetTexture(0,txt);

device.VertexFormat =
CustomVertex.PositionTextured .Format ;

device.DrawPrimitives(PrimitiveType.TriangleList, 0, 1);



device.EndScene ();
device.Present ();
angle += 0.05f;
}
```

```
protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}


protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

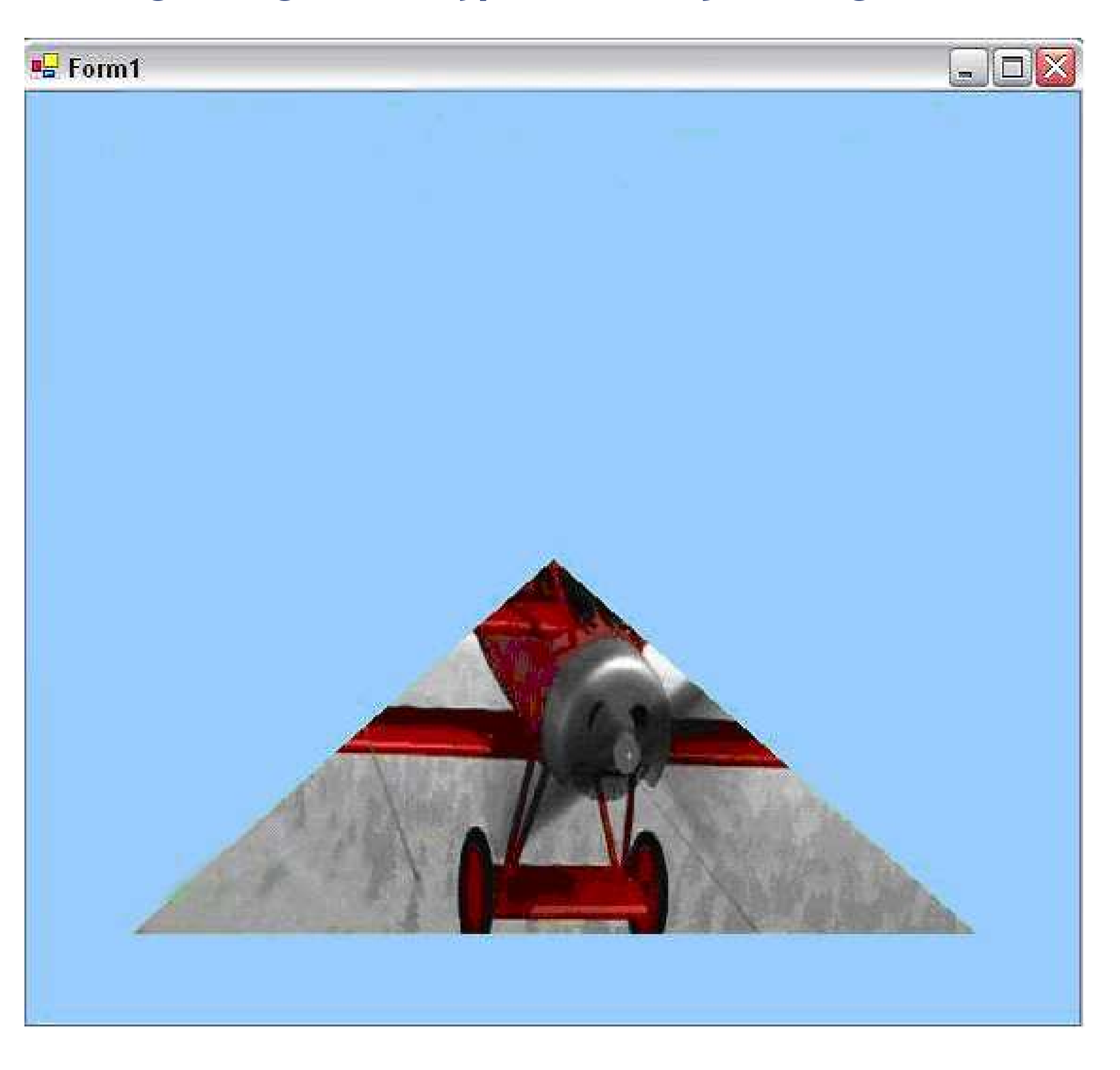
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
this.components = new
System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion


static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondeivce ();
xx.Show ();

while (xx.Created )
{
xx.camera ();
xx.position ();
xx.render ();
Application.DoEvents ();
}
}
```

```
            }
        }
    }
```

عند عمل RUN للكود بالأعلا فسيظهر عندنا الشكل بالأسفل:



لنأخذ مثال المكعب (الدرس الحادي عشر) ونقوم بعمل Texture له من وجهان..
كما في الشكل بالأسفل:



الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private Device device;
```

```
private System.ComponentModel.Container
components = null;
private float angle;
private VertexBuffer vb;
private Texture txt;



public Form1()
{
InitializeComponent();
}
public void ondeivce()
{
PresentParameters pp = new PresentParameters
();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat = DepthFormat.D16
;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);
txt =  TextureLoader.FromFile  (device,
Application.StartupPath + @"\..\..\RAAD.jpg");
}
public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,6),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.Transform.World =  Matrix.RotationAxis
(new Vector3 (1,2,3),angle);
device.RenderState.Lighting = false;
device.RenderState.CullMode = Cull.None;
}
public void position()
{

vb = new
VertexBuffer(typeof(CustomVertex.PositionTextured  ), 36, device,
Usage.SoftwareProcessing    |
Usage.WriteOnly,
CustomVertex.PositionTextured.Format, Pool.SystemMemory );

CustomVertex.PositionTextured  [] verts =
(CustomVertex.PositionTextured [])vb.Lock  (0,0);

//Front face
```

```
//المثلث الأول
verts[0].SetPosition (new Vector3 (-1.0f, 1.0f,
1.0f));
        verts[0].Tu = 0;
        verts[0].Tv = 0;

verts[1].SetPosition (new Vector3 (-1.0f, -1.0f,
1.0f));
        verts[1].Tu = 0;
        verts[1].Tv = 1;

verts[2].SetPosition (new Vector3 (1.0f, 1.0f,
1.0f));
        verts[2].Tu = 1;
        verts[2].Tv = 0;


//المثلث الثاني
verts[3].SetPosition (new Vector3 (-1.0f, -1.0f,
1.0f));
        verts[3].Tu = 0;
        verts[3].Tv = 1;

verts[4].SetPosition (new Vector3 (1.0f, -1.0f,
1.0f));
        verts[4].Tu = 1;
        verts[4].Tv = 1;

verts[5].SetPosition (new Vector3 (1.0f, 1.0f,
1.0f));
        verts[5].Tu = 1;
        verts[5].Tv = 0;


// Back face
//المثلث الأول
verts[6].SetPosition (new Vector3 (-1.0f, 1.0f, -
1.0f));
verts[7].SetPosition (new Vector3 (1.0f, 1.0f, -
1.0f));
verts[8].SetPosition (new Vector3 (-1.0f, -1.0f, -
1.0f));

//المثلث الثاني
verts[9].SetPosition (new Vector3 (-1.0f, -1.0f, -
1.0f));
verts[10].SetPosition (new Vector3 (1.0f, 1.0f, -
1.0f));
verts[11].SetPosition (new Vector3 (1.0f, -1.0f, -
1.0f));
```

```
// Top face
//المثلث الأول

verts[12].SetPosition (new Vector3 (-1.0f, 1.0f,
        1.0f));
                verts[12].Tu = 0;
                verts[12].Tv = 0;

verts[13].SetPosition (new Vector3 (1.0f, 1.0f, -
        1.0f));
                verts[13].Tu = 1;
                verts[13].Tv = 1;

verts[14].SetPosition (new Vector3 (-1.0f, 1.0f, -
        1.0f));
                verts[14].Tu = 0;
                verts[14].Tv = 1;
//المثلث الثاني

verts[15].SetPosition (new Vector3 (-1.0f, 1.0f,
        1.0f));
                verts[15].Tu = 0;
                verts[15].Tv = 0;

verts[16].SetPosition (new Vector3 (1.0f, 1.0f,
        1.0f));
                verts[16].Tu = 1;
                verts[16].Tv = 0;

verts[17].SetPosition (new Vector3 (1.0f, 1.0f, -
        1.0f));
                verts[17].Tu = 1;
                verts[17].Tv = 1;


// Bottom face
//المثلث الأول
verts[18].SetPosition (new Vector3 (-1.0f, -1.0f,
        1.0f));
verts[19].SetPosition (new Vector3 (-1.0f, -1.0f, -
        1.0f));
verts[20].SetPosition (new Vector3 (1.0f, -1.0f, -
        1.0f));

//المثلث الثاني
verts[21].SetPosition (new Vector3 (-1.0f, -1.0f,
        1.0f));
```

```
verts[22].SetPosition (new Vector3 (1.0f, -1.0f, -
1.0f));
verts[23].SetPosition (new Vector3 (1.0f, -1.0f,
1.0f));


// **** face
//المثلث الأول//
verts[24].SetPosition (new Vector3 (-1.0f, 1.0f,
1.0f));
verts[25].SetPosition (new Vector3 (-1.0f, -1.0f, -
1.0f));
verts[26].SetPosition (new Vector3 (-1.0f, -1.0f,
1.0f));

//المثلث الثاني//
verts[27].SetPosition (new Vector3 (-1.0f, 1.0f, -
1.0f));
verts[28].SetPosition (new Vector3 (-1.0f, -1.0f, -
1.0f));
verts[29].SetPosition (new Vector3 (-1.0f, 1.0f,
1.0f));


// Right face
//المثلث الأول//
verts[30].SetPosition (new Vector3 (1.0f, 1.0f,
1.0f));
verts[31].SetPosition (new Vector3 (1.0f, -1.0f,
1.0f));
verts[32].SetPosition (new Vector3 (1.0f, -1.0f, -
1.0f));

//المثلث الثاني//
verts[33].SetPosition (new Vector3 (1.0f, 1.0f, -
1.0f));
verts[34].SetPosition (new Vector3 (1.0f, 1.0f,
1.0f));
verts[35].SetPosition (new Vector3 (1.0f, -1.0f, -
1.0f));


vb.Unlock ();


}
public void render()
{
```

```
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.Black  ,1,1);
device.BeginScene ();

device.SetStreamSource(0, vb, 0);

device.SetTexture(0,txt);


device.VertexFormat =
CustomVertex.PositionTextured .Format ;

device.DrawPrimitives(PrimitiveType.TriangleList, 0, 12);


device.EndScene ();
device.Present ();
angle += 0.05f;
}
protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}


protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
this.components = new
System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
```

```
}
#endregion


static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondeivce ();
xx.Show ();

while (xx.Created )
{
xx.camera ();
xx.position ();
xx.render ();
Application.DoEvents ();
}
}
}
}
}
```

وبنفس النظرية لنطبقة على مثال الإسطوانه (الدرس الحادي), كما في الشكل بالأسفل:



الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

namespace WindowsApplication5
{

public class Form1 : System.Windows.Forms.Form
{
```

```
private float angle;
private Device device;
private VertexBuffer vb;
private Texture txt;
private System.ComponentModel.Container
components = null;

public Form1()
{
InitializeComponent();
}
public void ondevice()
{
PresentParameters pp = new PresentParameters
();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
pp.AutoDepthStencilFormat = DepthFormat.D16
;
pp.EnableAutoDepthStencil = true;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);
txt = TextureLoader.FromFile (device,
Application.StartupPath + @"\..\..\RAAD.jpg");
}
public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,-5),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.Transform.World = Matrix.RotationAxis
(new Vector3 (1,1,3),angle);
device.RenderState.Lighting = false;
device.RenderState.CullMode = Cull.None;
}
public void vertexbuffer()
{
vb = new VertexBuffer
(typeof(CustomVertex.PositionNormalTextured),100,device,Usage.Write
Only ,CustomVertex.PositionNormalTextured.Format ,Pool.Default );
CustomVertex.PositionNormalTextured [] cc =
(CustomVertex.PositionNormalTextured[])vb.Lock (0,0);
for (int i =0;i <50;i++)
{
float theta = (float)(3*Math.PI  *i)/60;
cc[i*2].SetPosition (new Vector3
((float)Math.Sin (theta),(float)Math.Cos (theta),0));
cc[i*2].SetNormal (new Vector3
((float)Math.Sin (theta),(float)Math.Cos (theta),0));
```

```
cc[i*2].Tu = ((float)i )/(49);
cc[i*2].Tv = 1;

cc[i*2 + 1].SetPosition (new Vector3
((float)Math.Sin (theta),(float)Math.Cos (theta),3));
cc[i*2 + 1].SetNormal (new Vector3
((float)Math.Sin (theta),(float)Math.Cos (theta),3));
cc[i*2 + 1].Tu = ((float)i )/(49);
cc[i*2 + 1].Tv = 0;
}
vb.Unlock ();
}
public void render()
{
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.Green  ,1,1);

device.BeginScene ();
device.SetStreamSource (0,vb,0);
device.SetTexture (0,txt);



device.VertexFormat =
CustomVertex.PositionNormalTextured.Format ;
device.DrawPrimitives
(PrimitiveType.TriangleStrip ,0,98);

device.EndScene ();
device.Present ();
angle += 0.05f;
}
protected override void OnPaint
(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}

protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}
```

```
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
this.components = new System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion


static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondevice ();
xx.Show ();
while (xx.Created )
{
xx.camera ();
xx.vertexbuffer ();
xx.render ();
Application.DoEvents ();
}

}
}
}
}
```

-أنواع الـVertex

نستطيع القول بأن هذا الموضوع مراجعة سريعة لما قد مر معنا من أنواع الـ Vertex

أولاًPositional Data :
والتي تعني توضع الإحداثيات على المحاور الثلاثة الـ x و الـ y والـz

Point ١ (٠،٠،٠)
Point ٢ (٠،١٠،٠)
Point ٣ (١٠،٠،٠)

(0,10,0)
Point 2

(0,0,0)
Point 1

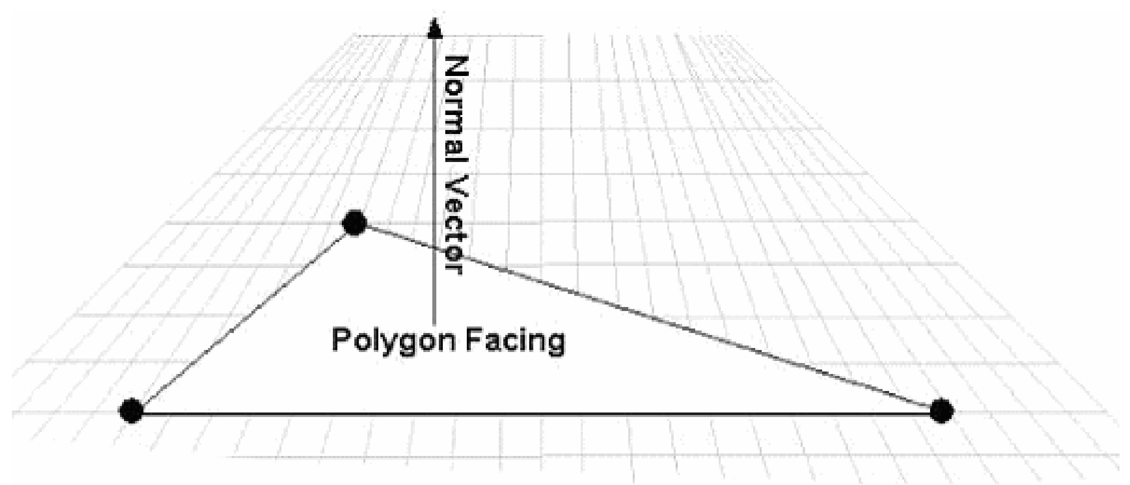
(10,0,0)
Point 3

Y
Z
X

نستخدم هذا النوع من أجل رسم الأشكال, وأيضاً من الممكن أن نضيف إلي هذه الـ Vertex خاصية الألوان.

ثانياً Normal Data :

وهو الخط الذي يكون عامودي على الـVertex



يستخدم هذا النوع عند التعامل مع الإضاءة ,(Lighting) سنتكلم عنه في

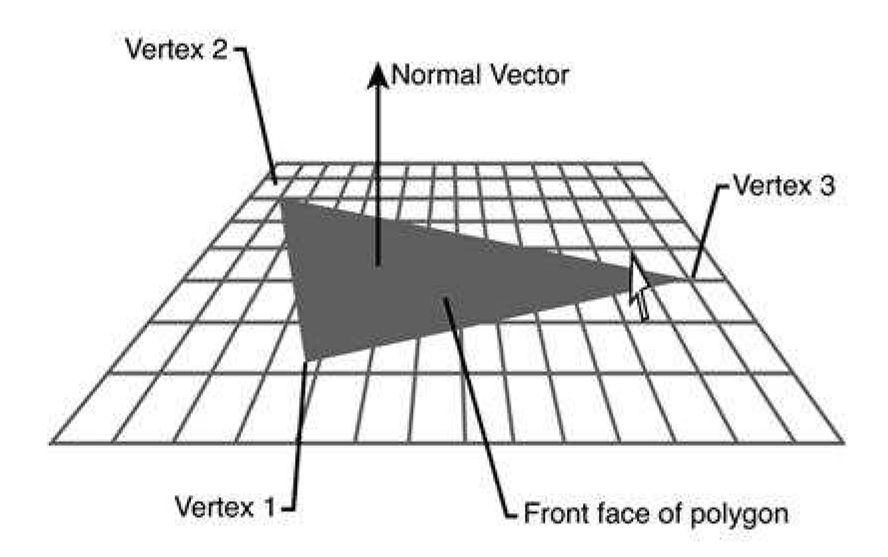


-الـLighting
-الـMesh

<u>-الـLighting</u>

يتيح لنا الـ Direct٣D ثلاثة أنواع من الإضاءه وهم:
الأول: الضوء المحيطي , (Ambient Light) يسقط هذا الضوء على كامل الجسم, حتى وإن لم تكن الإضاءة تقع مباشرة على الجسم) .يحدد كمية الضوء المنتشر التي ترتد عن السطح.(
الثاني: الضوء الإنتشاري ,(Diffuse Light) يسقط هذا الضوء في إتجاه متساوي على الأسطح (يحدد كمية الضوء المحيطي التي ترتد عن السطح.(
الثالث: الضوء الإنعكاسي ,(Specular Light) يسقط هذا الضوء على المنطقة المشار إليها فقط (يحدد كمية الضوء الإنعكاسي التي يعكسها السطح.(

تعتمد فكرة الإضاءة والظلال على حسابات الـ , Normal Vertex وقلنا سابقاً في الـ (Vertex Math) بأنه هو الخط العامودي من السطح يحسب هذا الخط عن طريق الـ Cross Product يكون توضعة كما في الشكل بالأسفل:



في عملية الإضاءة نستخدم نفس القاعدة ولأكن نقوم بعمل Normal Vector على نقاط الـ Vertex نفسها..

لنأخذ المثال التالي:
لنفرض أن لدي مثلت يحمل الإحداثيات التاليه:

كود:

```
verts[0].SetPosition(new Vector3(0.0f, 1.0f, 1.0f));
verts[1].SetPosition(new Vector3(-1.0f, -1.0f, 1.0f));
verts[2].SetPosition(new Vector3(1.0f, -1.0f, 1.0f));
```



ومن أجل عمل خط عامودي (Normal Vector) على تلك الإحداثيات نقوم بي إضافة الخاصية SetNormal لتأخذ نفس إحداثيات النقاط فبهذه العملية نحصل على زاوية ٩٠ درجة :

كود:

```
verts[0].SetNormal(new Vector3(0f, 1f, 1.0f));
verts[1].SetNormal(new Vector3(-1.0f, -1.0f, 1.0f));
verts[2].SetNormal(new Vector3(1.0f, -1.0f, 1.0f));
```

من رأيك الآن أن نرى كيف يتم حساب ذلك من المثال في الأعلا ... قانون الـ Corss Productهو التالي:

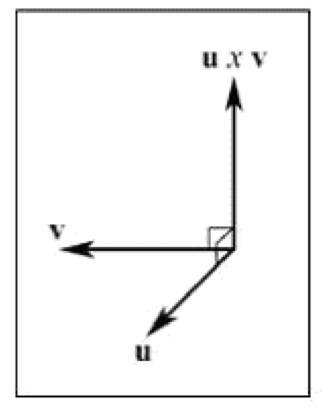
$$\mathbf{p} = \mathbf{u} \times \mathbf{v} = [(u_y v_z - u_z v_y), \quad (u_z v_x - u_x v_z), \quad (u_x v_y - u_y v_x)]$$

In component form:

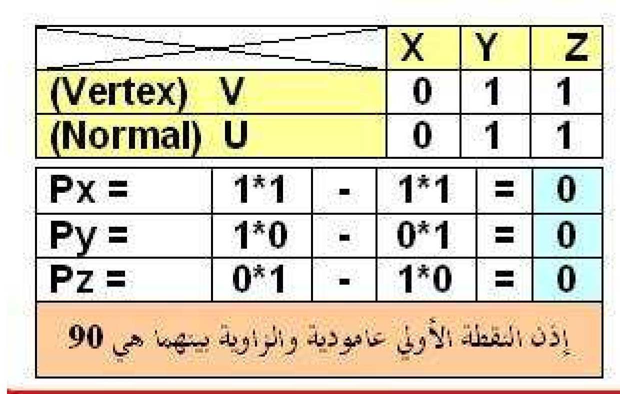
$$p_x = (u_y v_z - u_z v_y)$$

$$p_y = (u_z v_x - u_x v_z)$$

$$p_z = (u_x v_y - u_y v_x)$$



- If $\mathbf{u} \cdot \mathbf{v} = 0$, تكون الزاوية قائمة
- If $\mathbf{u} \cdot \mathbf{v} > 0$, تكون الزاوية حادة
- If $\mathbf{u} \cdot \mathbf{v} < 0$, تكون الزاوية منفرجة



	X	Y	Z
(Vertex) V	0	1	1
(Normal) U	0	1	1

Px =	1*1	-	1*1	=	0
Py =	1*0	-	0*1	=	0
Pz =	0*1	-	1*0	=	0

إذن النقطة الأولي عامودية والزاوية بينهما هي 90

	X	Y	Z
(Vertex) V	1-	1-	1
(Normal) U	1-	1-	1

Px =	-1*1	-	-1*-1	=	0
Py =	1*-1	-	-1*1	=	0
Pz =	-1*-1	-	-1*-1	=	0

إذن النقطة الثانية عامودية والزاوية بينهما هي 90

	X	Y	Z
(Vertex) V	1	1-	1
(Normal) U	1	1-	1

Px =	-1*1	-	1*-1	=	0
Py =	1*1	-	1*1	=	0
Pz =	1*-1	-	-1* 1	=	0

إذن النقطة الثالثة عامودية والزاوية بينهما هي 90

نأتي الآن إلى موضوع الـ Materials: يطلق هذا المصطلح عند عمل خصائص للضوء إذا إستخدمنا الـ Texture أو Mesh ونريد عمل إضاءة عليها .. فيجب أن نستخدم هذا المصطلح....

التصريح عن الـ Material يكون كالتالي:
أولاً: عمل كائن (Object) للضوء.
ثانياً: عمل كائن (Object) للـMaterial
ثالثاً: إعطاء لون للـDiffuse
رابعاً: إعطاء لون للـAmbient
خامساً: تعريف الـ device على الـMaterial

كود:

```
Color col = Color.White   ;
```

```
Material mtrl = new Material();
mtrl.Diffuse = col;
mtrl.Ambient = col;
device.Material = mtrl;
```

طبعاً: هذه الخطوات ليست ثابته, فإنك تستطيع تغيرها كما تشاء بحسب الضوء الذي تريده .

لنراجع ما مر معنا إلى الآن ...

كما تذكر بدأنا في أول الدرس بالتكلم عن الـ Normal Vertex والذي يساعدنا في عملية حسابات الضوء, ومن ثم أخذنا الحديث إلى الـ Material والتي تصف خصائص الضوء إذا كانت هناك Texture أو Mesh..... والآن جاء دور الخطوة الأخيرة وهي التصريح عن الضوء نفسه.

أولاً: نقوم بالتصريح على الضوء بداخل مصفوفة..
أي أن

كود:

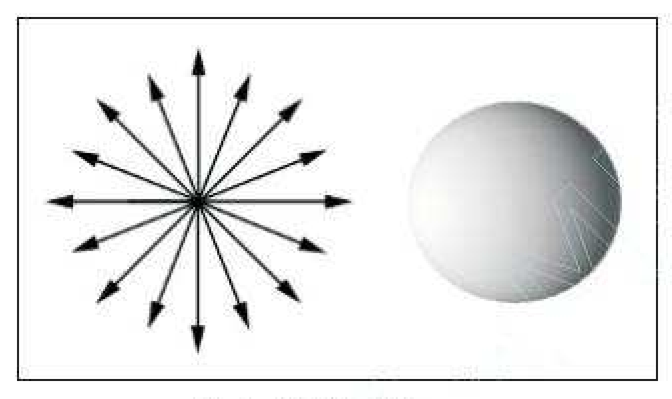
```
device.Lights[0]
```

يمثل الضوء الأول..

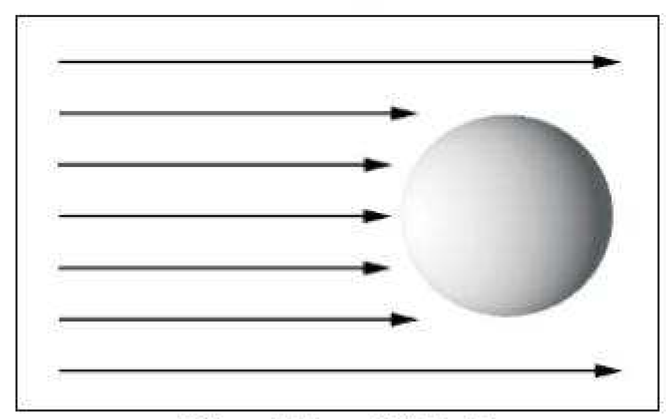
كود:

```
device.Lights[1]
```

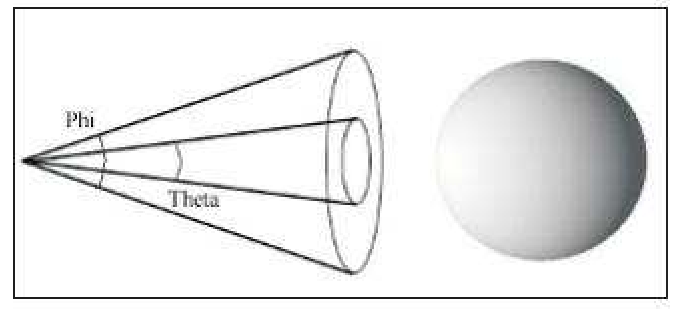
يمثل الضوء الثاني..
وهكذا....
ثانياً: طريقة عرض الضوء: وتوجد ثلاثة أنواع هي كما في الشكل بالأسفل:



Point Light



Directional Light



Spot Light

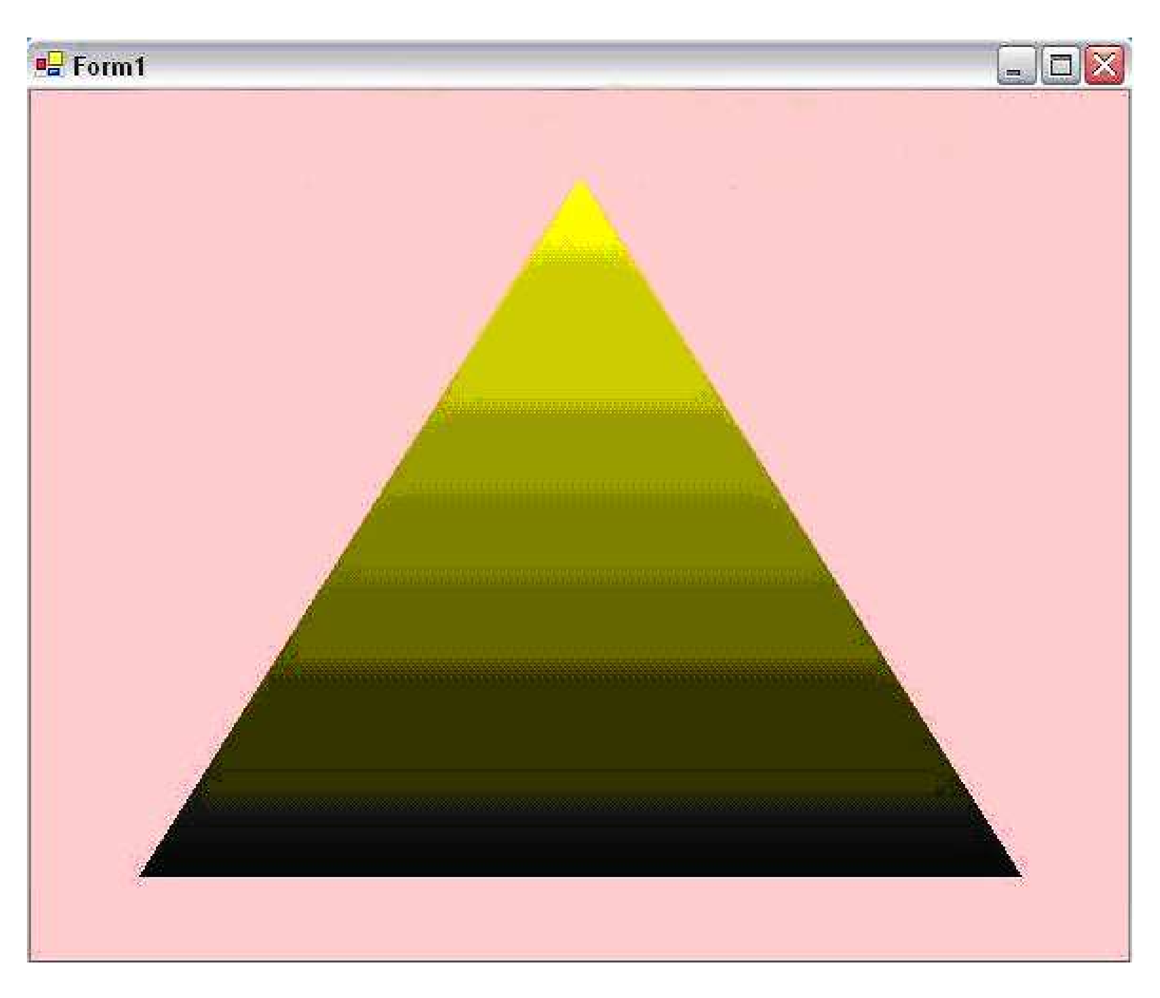
ثالثاً :تحديد نوع اللون المنتشر على السطح..
رابعاً: نخزن (Commit) هذا اللون بذاكرة كرت الشاشة .
خامساً: نجعل خاصية الـ Enable تساوي True ليتسنى له معالجة الإضاءة.

كود:

```
device.Lights[0].Type = LightType.Directional;
device.Lights[0].Diffuse = System.Drawing.Color.White  ;
device.Lights[0].Direction = new Vector3();
```

```
device.Lights[0].Commit();
device.Lights[0].Enabled = true;
```

الآن لنبدأ بي المثال الأول: وهو كما في الشكل بالأسفل...



حيث قمنا بعمل إضاءة للنقطة التي في رأس المثلث, وجعلنا الإضاءة فيها باللون الأصفر ..

الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace WindowsApplication6
{
```

```
public class Form1 : System.Windows.Forms.Form
{
private Device device;
private System.ComponentModel.Container
components = null;
private float angle;
private VertexBuffer vb;



public Form1()
{
InitializeComponent();
}
public void ondeivce()
{
PresentParameters pp = new PresentParameters
();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat = DepthFormat.D16
;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);
}
public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,4),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.RenderState.Lighting = true;
device.RenderState.CullMode = Cull.None;
}
public void light()
{

/*Color col = Color.Black       ;

Material mtrl = new Material();

mtrl.Diffuse = col;
mtrl.Ambient = col;
device.Material = mtrl;
*/

device.Lights[0].Type = LightType.Directional;
device.Lights[0].Diffuse =
System.Drawing.Color.Yellow      ;
```

```
//device.Lights[0].Direction = new
Vector3((float)Math.Cos(Environment.TickCount / 250.0f), 0,
(float)Math.Sin(Environment.TickCount / 250.0f));
device.Lights[0].Direction = new Vector3(0,-1,-1);
device.Lights[0].Commit();//let d3d know about
the light
device.Lights[0].Enabled = true;//turn it on



}
public void position()
{

vb = new
VertexBuffer(typeof(CustomVertex.PositionNormalColored   ), 3, device,
Usage.SoftwareProcessing    |
Usage.WriteOnly,
CustomVertex.PositionNormalColored .Format, Pool.SystemMemory );

CustomVertex.PositionNormalColored   [] verts
= (CustomVertex.PositionNormalColored  [])vb.Lock  (0,0);

verts[0].SetPosition(new Vector3(0.0f, 1.0f,
1.0f));
verts[0].SetNormal(new Vector3(0f, 1f, 1.0f));
verts[0].Color = System.Drawing.Color.White
.ToArgb();

verts[1].SetPosition(new Vector3(-1.0f, -1.0f,
1.0f));
//verts[1].SetNormal(new Vector3(-1.0f, -1.0f,
1.0f));
verts[1].Color = System.Drawing.Color.White
.ToArgb();

verts[2].SetPosition(new Vector3(1.0f, -1.0f,
1.0f));
//verts[2].SetNormal(new Vector3(1.0f, -1.0f,
1.0f));
verts[2].Color = System.Drawing.Color.White
.ToArgb();



vb.Unlock ();



}
public void render()
```

```
{
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.LightPink  ,1,1);
device.BeginScene ();

device.SetStreamSource(0, vb, 0);

device.VertexFormat =
CustomVertex.PositionNormalColored  .Format ;

device.DrawPrimitives(PrimitiveType.TriangleList, 0, 1);



//device.Transform.World =  Matrix.RotationAxis
(new Vector3 (1,1,0),angle);


device.EndScene ();
device.Present ();
angle += 0.05f;
}
protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}


protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
this.components = new
System.ComponentModel.Container();
```

```
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion


static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondeivce ();
xx.Show ();

while (xx.Created )
{
xx.camera ();
xx.position ();
xx.light ();
xx.render ();
Application.DoEvents ();
}
}
}
}
}
```

هنا لدينا بعض الملاحظات على الكود بالأعلا...
أولاً: عند إعطاء لون للـ Vertex نجعلها بلون خفيف مثل الأبيض, لكي يعطي الضوء تأثيرة.
ثانياً: العلاقة ما بين إحداثيات الـ SetPosition و SetNormal وDirection

كود:

```
verts[0].SetPosition(new Vector3(0.0f, 1.0f, 1.0f));
```

كود:

```
verts[0].SetNormal(new Vector3(0f, 1f, 1.0f));
```

كود:

```
device.Lights[0].Direction = new Vector3(0,-1,-1);
```

يأتي أول شئ الـ SetPosition والذي يعمل على تحديد الإحداثيات للنقطة, يأتي بعدها الـ SetNormal لعمل خط عامودي على هذه النقطة من أجل عكس الإضاءة ويمكننا إعطائها أي إحداثيات ... وفي مثالنا هذا قمنا بإعطاءها نفس إحداثيات الـ SetPosition لتكون الزاوية بينهما ٩٠ وليسهل التعامل معها .. وتكلمنا عن هذا بالأعلا .. نأتي إلى الخطوة الأخيرة وهي تحديد إحداثيات الـ Direction للإضاءة بالإعتماد على Normal Vertex أي إحداثيات الـ SetNormal وبما أننا نعمل بي قاعدة الـ Hand) **** راجع درس الـ



(Cameraأي مع عقارب الساعة, لذلك سيكون الـ Normal Vertex من أسفل النقطة, لذلك نأخذ معكوس النقاط لإسقاط الضوء ...
مختصر الكلام ... ضع الـ SetNormal مثل إحداثيات الـ SetPosition وإعكسها عند الـ Direction أي إجعل الموجب سالب والسالب موجب ..

-الـMesh

أصبحنا نعلم الآن أن أي جسم ما هو سوى نقاط Vertex متصلة ببعضها البعض بواسطة خطوط مثل الشبكة , لتكون في النهاية الجسم.
وكما تذكر في الدروس السابقة قمنا برسم أشكال بسيطة مثل المثلث والمكعب والإسطوانة ... حسناً الآن ماذا عن الأشكال المعقدة من وجه شخص أو طائرة أو سيارة.. هنا جاء مصطلح الـ Mesh ليحل لنا هذه المشكلة وليسهل علينا العمل .
الـ Mesh هو عبارة عن ملف يسمى بي الـ XFile وذلك لأن إمتداد الملف (Extension)هو , X. بدأت إستخدام هذا التقنية من النسخة السادسة للـ DirectXوتقوم فكرتها على التالي: عند رسم أي شكل أو جسم في البرامج مثل ٣D Max أو الـ Maya أو برامج معالجة الأشكال ثلاثية الأبعاد, فإننا نقوم بي تخزينها بصيغة X File ليتسنى لنا بعد ذلك قراءة هذا الملف بواسطة الـ DirectX

الآن لنأخذ المثال التالي:
وهو عبارة عن سيارة ... تستطيع تحميل الملف من هنـــا , بعد تنزيل الملف والذي يحمل الإسم , CAR أولاً: نقوم بوضعه بداخل ملف المشروع.
ثانياً: نضيف الكود التالي في منطقة التعريفات:
private Mesh mesh;



ثالثاً: نقوم بعمل (Function) خاصة بي الـ mesh لنقوم بتعريف الـ Mesh بداخلها وتكون كالتالي:

كود:

```
public void meshh()
{

mesh = Mesh.FromFile (Application.StartupPath +
@"\..\..\CAR.x",MeshFlags.SystemMemory ,device);

}
```

Mesh.FromFileتستخدم هذه الـ function من أجل تحديد مسار ملف الـ Meshحيث حددناه بأنه بداخل نفس ملف المشروع بواسطة Application.StartupPathومن ثم حددنا إسم الملف وهو.CAR.X
نأتي الآن إلى الـ Parameter الآخر وهو الـ MeshFlags والذي يحدد أين سوف يتم تخزين بيانات (Data) الخاصة بي الملف Car حيث قمنا في الكود بالأعلا بإختيار الـ SystemMemory والتي تعني بأن عملية تخزين بيانات الملف ستتم بداخل ذاكرة الـ RAM الخاصة بي الكمبيوتر... ولمزيد من المعلومات حول الإختلافات بين هذه الخيارات أنظر إلى الجدول بالأسفل.

خطأ!

MeshFlags.DoNotClip	Use the Usage.DoNotClip flag for vertex and index buffers.
MeshFlags.Dynamic	Equivalent to using both IbDynamic and VbDynamic.
MeshFlags.IbDynamic	Use Usage.Dynamic for index buffers.
MeshFlags.IbManaged	Use the Pool.Managed memory store for index buffers.
MeshFlags.IbSoftwareProcessing	Use the Usage.SoftwareProcessing flag for index buffers.
MeshFlags.IbSystemMem	Use the Pool.SystemMemory memory pool for index buffers.
MeshFlags.IbWriteOnly	Use the Usage.WriteOnly flag for index buffers.
MeshFlags.VbDynamic	Use Usage.Dynamic for vertex buffers.
MeshFlags.VbManaged	Use the Pool.Managed memory store for vertex buffers.
MeshFlags.VbSoftwareProcessing	Use the Usage.SoftwareProcessing flag for vertex buffers.
MeshFlags.VbSystemMem	Use the Pool.SystemMemory memory pool for vertex buffers.
MeshFlags.VbWriteOnly	Use the Usage.WriteOnly flag for vertex buffers.
MeshFlags.Managed	Equivalent to using both IbManaged and VbManaged.
MeshFlags.Npatches	Use the Usage.NPatches flag for both index and vertex buffers. This is required if the mesh will be rendered using N-Patch enhancement.
MeshFlags.Points	Use the Usage.Points flag for both index and vertex buffers.
MeshFlags.RtPatches	Use the Usage.RtPatches flag for both index and vertex buffers.
MeshFlags.SoftwareProcessing	Equivalent to using both IbSoftwareProcessing and VbSoftwareProcessing.
MeshFlags.SystemMemory	Equivalent to using both IbSystemMem and VbSystemMem.
MeshFlags.Use32Bit	Use 32-bit indices for the index buffer. While possible, normally not recommended.
MeshFlags.UseHardwareOnly	Use hardware processing only.

بقي لدينا الـ Parameter الأخير وهو الخاص بي الكائن object لكرت الشاشة والذي أطلقنا عليه الإسمdevice

رابعاً: بقيت الآن مرحلة الرسم, أي إرسال الملف (CAR) من الـ RAM إلى كرت الشاشة لقراءته, في الحقيقة إن عملية إرسال البيانات من الـ RAM إلى كرت الشاشة لا تتم بدفعة واحدة, وإنما تتم عن طريق تقسيم الملف CARإلى أجزاء تسمى بي SubSet, لذلك نقوم بي إستخدام الـ For من أجل تحميل هذه الأجزاء ومن ثم قرائتها ورسمها بواسطة الدالة (function) المسمى .DrawSubset

كود:

```
for (int i=0;i < 20 ;i++)
{
    mesh.DrawSubset (i);
}
```

إذا أردت أن تسألني كم حجم كل subset فجوابي سيكون ... الله أعلم ولأكنه بالتأكيد سيعتمد على حجم الـ RAM وسرعة الجهاز ونوع الـ VGA

بالإضافة إلى حجم الملف نفسه.

الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private Device device;
private float angle;
private Mesh mesh;
private System.ComponentModel.Container
components = null;

public Form1()
{
InitializeComponent();
}
public void ondevice()
{
PresentParameters pp = new PresentParameters
();
pp.SwapEffect = SwapEffect.Discard;
pp.Windowed = true;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat = DepthFormat.D16
;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);

}
public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,-20),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.Transform.World = Matrix.RotationAxis
(new Vector3 (0,3,0),angle) * Matrix.Scaling (1,1,1) ;
```

```
device.RenderState.CullMode = Cull.None;
device.RenderState.Lighting =false;

}

public void meshh()
{

mesh = Mesh.FromFile  (Application.StartupPath
+ @"\..\..\CAR.x",MeshFlags.SystemMemory ,device);

}

public void render()
{
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.Blue ,1,1);

device.BeginScene ();

for (int i=0;i < 20 ;i++)
{
mesh.DrawSubset (i);
}

device.EndScene ();
device.Present ();
angle += 0.05f;
}
protected override void OnPaint
(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}



protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

#region Windows Form Designer generated code
```

```
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
this.components = new
System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion

static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondevice ();
xx.Show ();

while (xx.Created )
{
xx.camera ();
xx.meshh ();
xx.render ();
Application.DoEvents ();
}
}

}
}
}
```

ما رأيك أن نأخذ فكرة أكبر عن ملفات الـ X , قم بفتح الملف CAR التابع للمثال بالأعلا عن طريق الـ Notepad (المفكره) لنلاحظ التالي:

عند فتح هذا الملف فإن أول سطر منه يحوي التالي:
Xofوالتي تمثل الحروف المميزة للـ xfile ليستطيع الـ DirectX التعرف علية.
٠٣والتي تمثل رقم النسخة.
Txtوالتي تمثل نوع الملف.
٠٠٣٢والتي تمثل نوع البيانات.

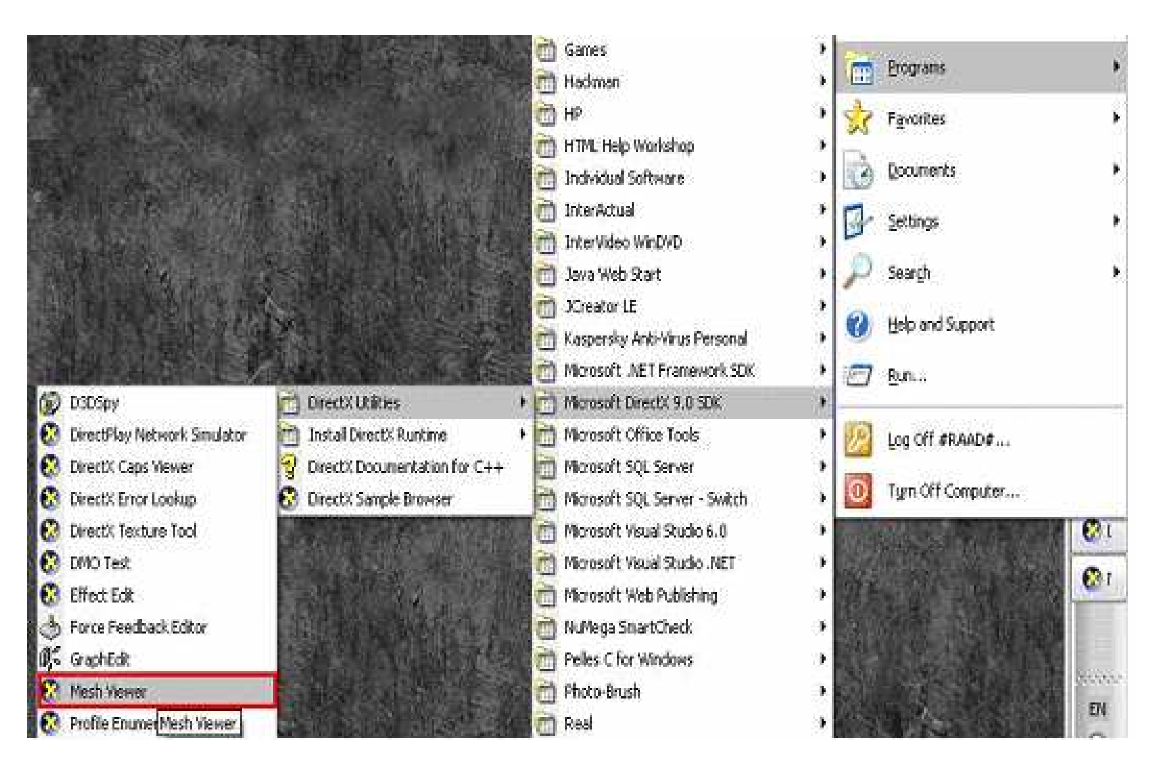
وأما باقي الملف فهو عبارة عن قوالب (Template), كل قالب يحتوي على أربع أقسام هي:
القسم الأول: يصف نوع القالب ,MeshNormals, Texture, Vertex)
الخXSkinMeshHeader) ...
القسم الثاني: رقم خاص بكل قالب مثل-٩٣c٠-٤٤ab-ff٧c-٣cf١٦٩ce>)

f٧٨f٦٢d١٧٢e٢>) يسمى هذا الرقم بي Universally Unique Identifier
ويختصر إلى.(UUID)
القسم الثالث: يحوي على مكونات القالب اي الـ Datatype
القسم الرابع: تكون من أجل عملية التحكم بي فتح وإغلاق القالب وتأخذ الشكل التالي.[...]

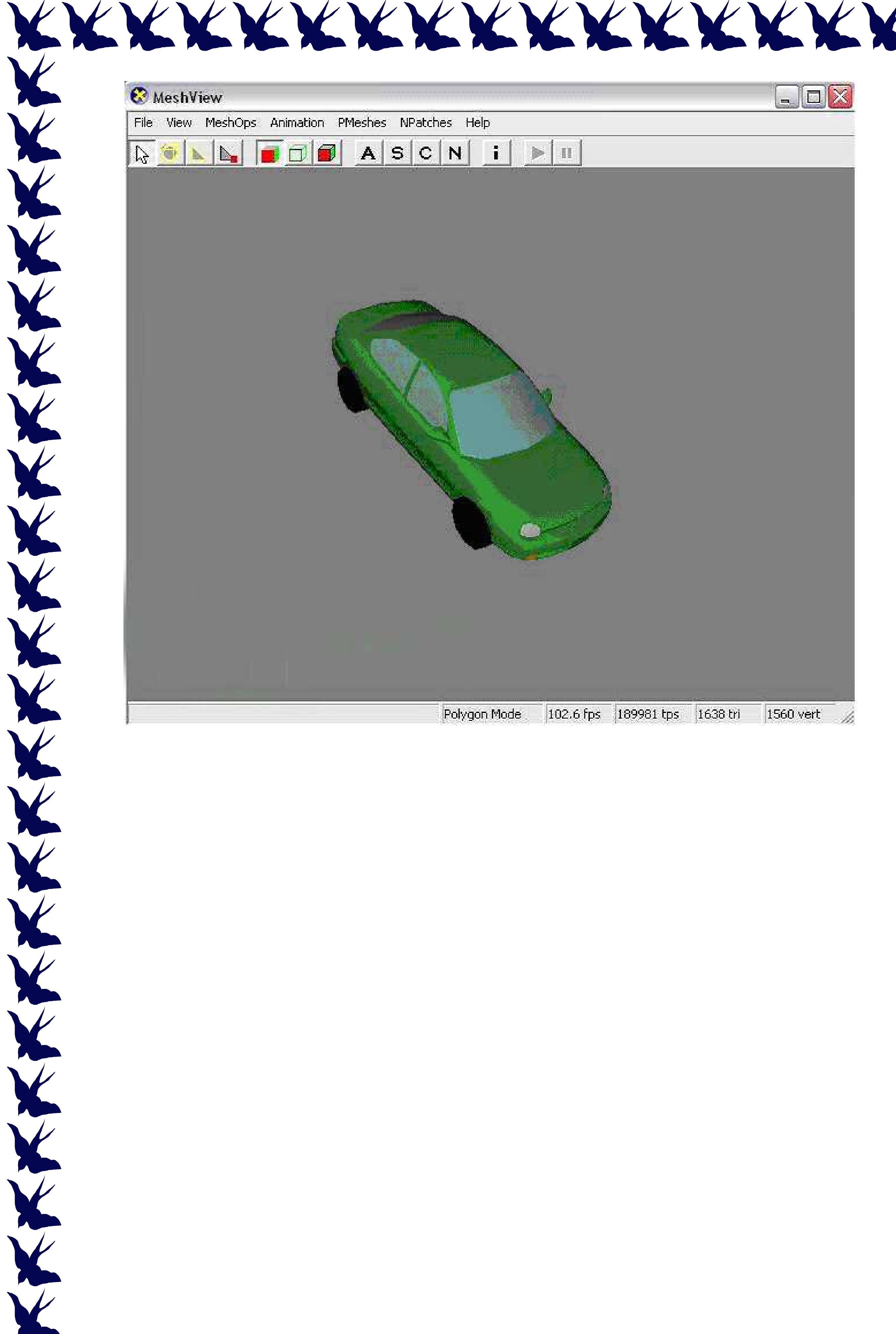
)ينطبق هذا الكلام على جميع ملفات الـX)

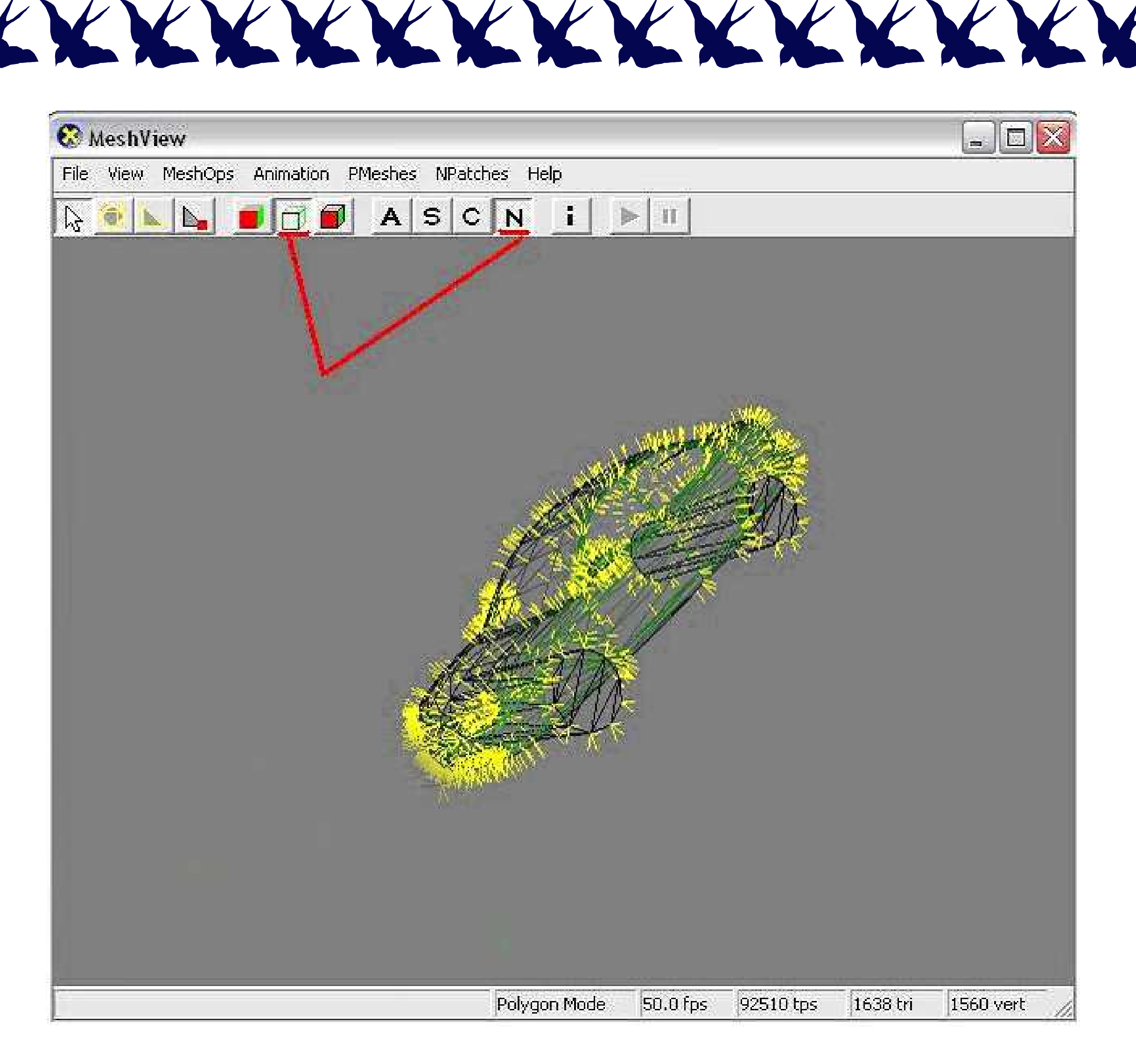
نستنتج من الكلام بالأعلا, بأنني من الممكن أن أعالج جميع الـ Matrix والـNormal Vectors والـ Texture والـ Vertex بواسطة برنامج مثل الـ ٣ D MAX وأخزنها بصيغة الـ X FILE ويبقى على الـ DirectX قراءة هذا الملف فقط ... 🙂

أتريد إثبات لهذا الكلام ... حسناً. إفتح الـ Mesh Viewer وهو إحدى البرامج الإضافية التي تنزل مع الـ DirectX SDK كما في الشكل بالأسفل..



بعدما ينفتح البرنامج إذهب إلى File ومن ثم Open Mesh File إختر مكان الملف.. CAR
لتظهر لنا السيارة بالشكل التالي:


MeshView
File View MeshOps Animation PMeshes NPatches Help
A S C N
Polygon Mode
102.6 fps
189981 tps
1638 tri
1560 vert



نستنتج من هذان الشكلان بالأعلا بأن هذا الـ Mesh يحتوي على الألوان وعلى الـ) Normal Vector وهي الخطوط الصفراء في الشكل الثاني, تستخدم لعكس الإضاءة.)

إذن فالشخص الذي رسم هذه بالسيارة بواسطة برنامج الـ ٣ max قام بوضع الألوان وتحديد نقاط الـ Normal, وبهذا العمل فقد أراحنا من عناء من معالجة هذه الأشياء بواسطة الـ DirectX

ولأكن السؤال هنا.... لماذا لم يظهر الـ DirectX الألوان للسيارة مع أنها موجودة بداخل الـ Mesh ؟؟ أجبنا على هذا السؤال في الأعلا عندما قلنا بأنه عندما نريد التعامل مع الـ Texture أو الـ Mesh فيجب علينا عمل .Material نتبع الخطوات التالية:
نبدأ بالتصريح عن الـ Material كالتالي:

كود:

```
private Material[] mat;
```

ثانياً :نقوم بعمل Object للمصفوفة ExtendedMaterial حيث تعمل هذه المصفوفة على تخزين معلومات الـ Subset الخاصة بي الـMesh

كود:

```
ExtendedMaterial[] material;
```

ثالثاً :تحديد الملف وطريقة تخزينة, (تكلمنا عنه في الأعلا) ولأكن يزيد هذا الكود في الجملة التالية out material والتي تعمل على تفريغ كل Subset بداخل الـ ExtendMaterial ومن ثم العودة.

كود:

```
mesh = Mesh.FromFile (Application.StartupPath +
@"\..\..\CAR.x",MeshFlags.SystemMemory ,device,out material);
```

رابعاً: عمل for loop للـ subset بحسب عددها.material.Length
نحدد نوع الـ Material بي أنها.Material٣D
نصرح عن الـ Ambient و الـ Diffuse من أجل عملية إعطاء الإضاءة.

كود:

```
for (int i= 0;i < material.Length ;i++)
{
    mat[i] = material[i].Material3D ;
    mat[i].Ambient = mat[i].Diffuse ;
}
```

المرحلة الآخيرة وهي قراءة (رسم) الـ Subset والـMaterials

كود:

```
for (int i=0;i < mat.Length ;i++)
{
        device.Material = mat[i];
        mesh.DrawSubset (i);
}
```

خطأ!



الكود كاملاً :

حيث جعلنا الإضاءة بالألون الأبيض لذلك سيظهر الشكل باللون أفتح.

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private Device device;
private float angle;
private Mesh mesh;
private Material[] mat;
```

```
private System.ComponentModel.Container
components = null;

public Form1()
{
InitializeComponent();
}
public void ondevice()
{
PresentParameters pp = new PresentParameters
();
pp.SwapEffect = SwapEffect.Discard;
pp.Windowed = true;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat = DepthFormat.D16
;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);

}
public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,-20),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.Transform.World = Matrix.RotationAxis
(new Vector3 (0,3,0),angle) * Matrix.Scaling (1,1,1) ;

device.RenderState.CullMode = Cull.None;
device.RenderState.Lighting =true;
device.RenderState.Ambient = Color.White     ;
}
public void meshh()
{
ExtendedMaterial[] material;

mesh = Mesh.FromFile (Application.StartupPath
+ @"\..\..\CAR.x",MeshFlags.SystemMemory ,device,out material);
mat = new Material [material.Length];
for (int i= 0;i < material.Length ;i++)
{
mat[i] = material[i].Material3D   ;
mat[i].Ambient = mat[i].Diffuse ;
}
}
public void light()
{
Color col = Color.White ;
Material mm = new Material ();
mm.Ambient = col;
```

```
mm.Diffuse = col;
device.Material = mm;

device.Lights [0].Diffuse = Color.White;
device.Lights [0].Type = LightType.Directional ;
device.Lights [0].Direction = new Vector3 (0,0,2);
device.Lights [0].Commit ();
device.Lights [0].Enabled = true;
}
public void render()
{
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.Blue ,1,1);

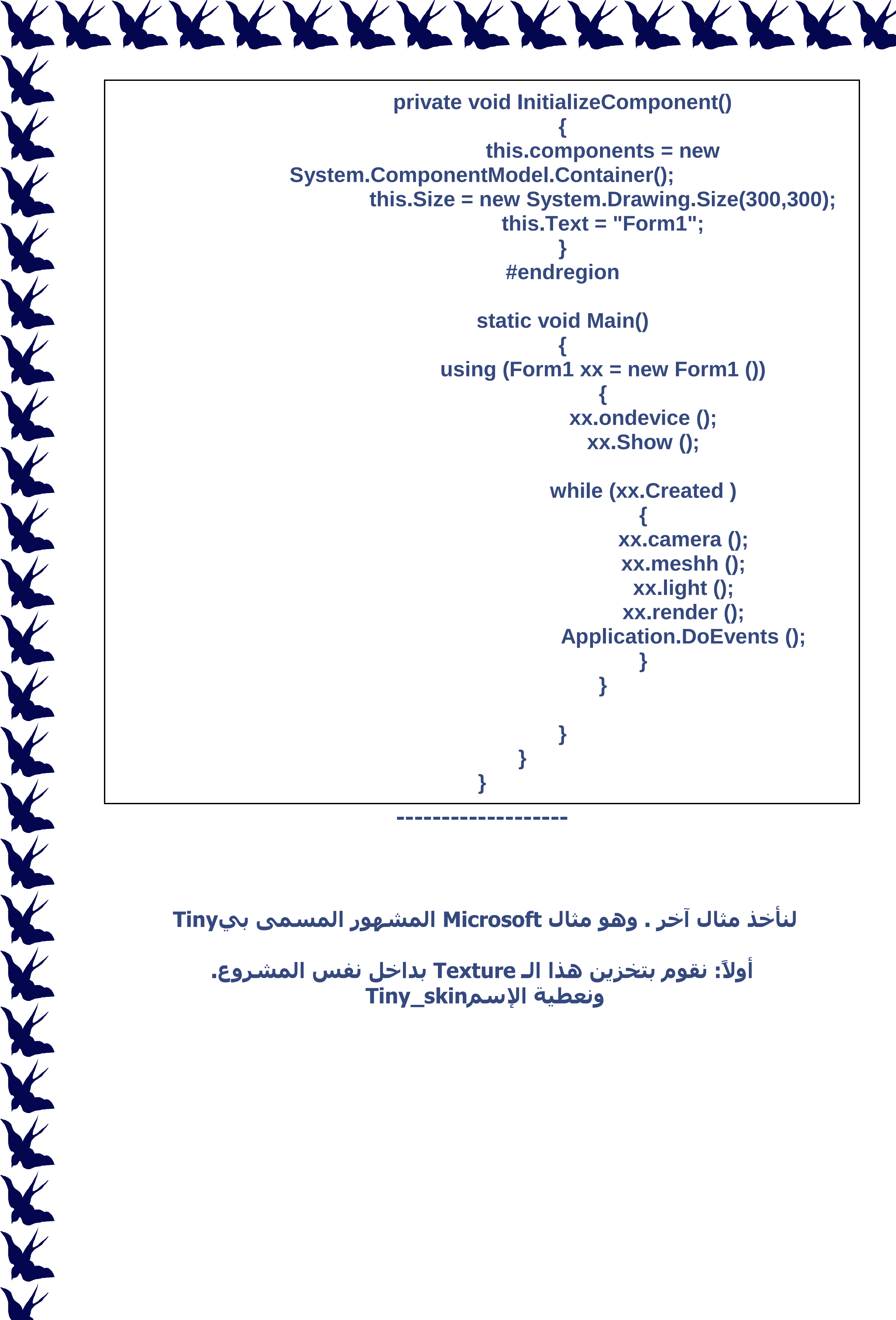
device.BeginScene ();

for (int i=0;i < mat.Length ;i++)
{
device.Material = mat[i];
mesh.DrawSubset (i);
}
device.EndScene ();
device.Present ();
angle += 0.05f;
}
protected override void OnPaint
(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}



protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
```



```
private void InitializeComponent()
{
this.components = new
System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion

static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondevice ();
xx.Show ();

while (xx.Created )
{
xx.camera ();
xx.meshh ();
xx.light ();
xx.render ();
Application.DoEvents ();
}
}

}
}
}
```

لنأخذ مثال آخر . وهو مثال Microsoft المشهور المسمى بيTiny

أولاً: نقوم بتخزين هذا الـ Texture بداخل نفس المشروع.
ونعطية الإسمTiny_skin



ثانياً :نقوم بوضع هذا الـ Mesh أيضاً بداخل نفس المشروع ونعطية الإسم tiny

ومن ثم نستخدم نفس الخطوات التي تكلمنا عليها بمثال السيارة ... لعمل Material

الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private Device device;
private float angle;

private Mesh mesh;
private Material[] mat;
private Texture txt;
private System.ComponentModel.Container
components = null;

public Form1()
{
```

```
InitializeComponent();
}
public void ondevice()
{
PresentParameters pp = new PresentParameters
();
pp.SwapEffect = SwapEffect.Discard;
pp.Windowed = true;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat = DepthFormat.D16
;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);
txt = TextureLoader.FromFile
(device,Application.StartupPath + @"\..\..\Tiny_skin.bmp");
}
public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height
,1,2000);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (1,-1000,30),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.Transform.World = Matrix.RotationAxis
(new Vector3 (0,0,2),angle) * Matrix.Scaling (1,1,1) ;

device.RenderState.CullMode = Cull.None;
device.RenderState.Lighting =true;
device.RenderState.Ambient = Color.White;
}
public void meshh()
{
ExtendedMaterial[] material;
Directory.SetCurrentDirectory
(Application.StartupPath + @"\..\..\");
mesh = Mesh.FromFile
("Tiny.x",MeshFlags.SystemMemory ,device,out material);
mat = new Material [material.Length];
for (int i= 0;i < material.Length ;i++)
{
mat[i] = material[i].Material3D ;
mat[i].Ambient = mat[i].Diffuse ;
}
}
public void light()
{
Color col = Color.White ;
Material mm = new Material ();
mm.Ambient = col;
mm.Diffuse = col;
device.Material = mm;
```

```
device.Lights [0].Diffuse = Color.White;
device.Lights [0].Type = LightType.Directional ;
device.Lights [0].Direction = new Vector3 (0,0,6);
device.Lights [0].Commit ();
device.Lights [0].Enabled = true;
}
public void render()
{
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.Blue ,1,1);
device.SetTexture (0,txt);
device.BeginScene ();

for (int i=0;i < mat.Length ;i++)
{
device.Material = mat[i];
mesh.DrawSubset (i);
}
device.EndScene ();
device.Present ();
angle += 0.5f;
}
protected override void OnPaint
(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}



protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

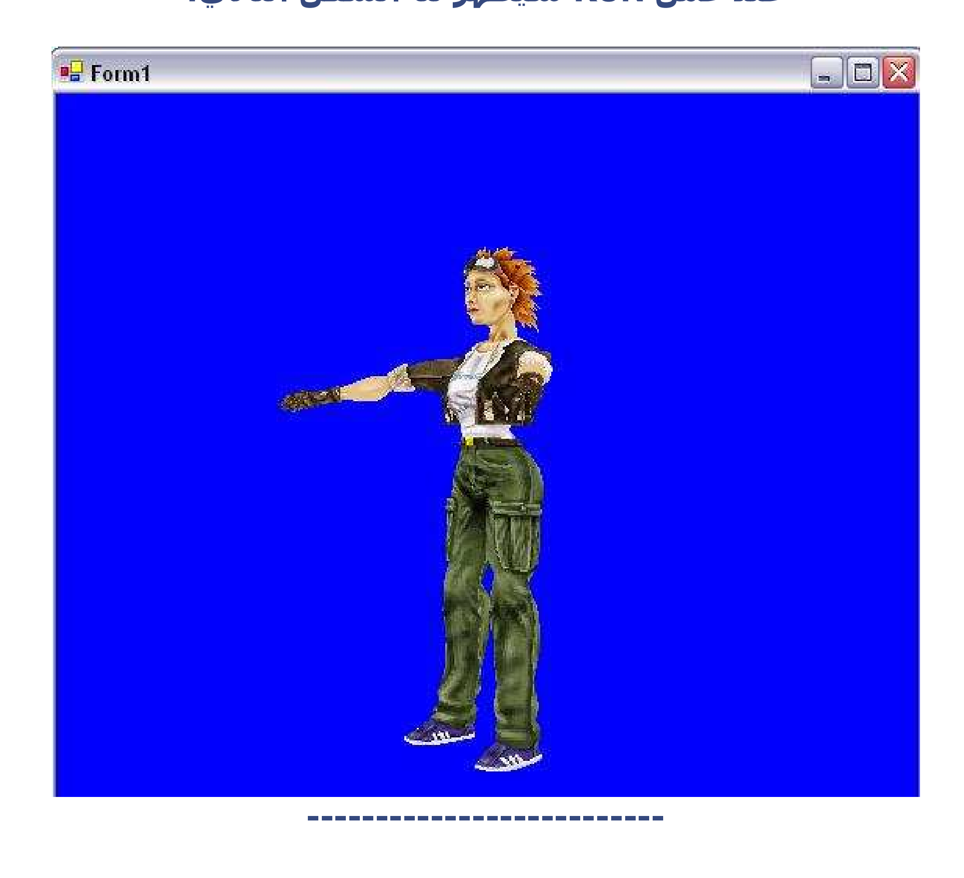
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
```

```
this.components = new
System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion

static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondevice ();
xx.Show ();

while (xx.Created )
{
xx.camera ();
xx.meshh ();
xx.light ();
xx.render ();
Application.DoEvents ();
}
}
}
}
}
```

عند عمل RUN سيظهر لنا الشكل التالي:





الدرس الرابع عشر : (الأخير)

-تحريك الأشكال
-الـFont

-تحريك الأشكال

سوف نأخذ هنا تحريك الجسم بأبسط حالاته وهي (يمين, يسار, اعلى, اسفل,قريب, بعيد) سنطبق هذا الكلام على مثال السيارة الذي تكلمنا عنه في الدرس الماضي:
أولاً: نقوم بعمل ثلاثة متغيرات وإعطائها قيمة إبتدائية ٠,٧ تشير هذه المتغيرات إلى المحاور X و Y وZ

كود:
```
float posx = 0.7f;
float posy = 0.7f;
float posz = 0.7f;
```

ثانياً: نقوم بوضع هذه المتغيرات بداخل الـ Matrix Translation) أعتقد بأن هذا الكلام واضح ففي الدروس السابقة شرحنا ما الذي تقوم به هذه الـ Matrix)

كود:
```
Matrix.Translation (posx,posy,posz)
```

ثالثاً :إستدعاء الدالة OnKeyDown من أجل تسجيل أي ضغطة على الـ keyboard

كود:
```
case Keys.Right :
posx ++;
break;
```

عند الضغط على زر (السهم الأيمن) سيعمل على تحريك الجسم بشكل تزايدي إلى الجهه اليمنى للمحور x, وذلك بحسب الـMatrix.Translation

كود:
```
protected override void OnKeyDown
(System.Windows.Forms.KeyEventArgs e)
{
    switch (e.KeyCode )
    {
        case Keys.Right :
            posx ++;
            break;

        case Keys.**** :
```



```
posx --;
break;

case Keys.Up :
posy ++;
break;

case Keys.Down :
posy --;
break;

case Keys.Home :
posz ++;
break;

case Keys.End :
posz --;
break;

}


}
```

الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private Device device;
private float angle;
private Mesh mesh;
private Material[] mat;
```

```
private System.ComponentModel.Container
components = null;
float posx = 0.7f;
float posy = 0.7f;
float posz = 0.7f;

public Form1()
{
InitializeComponent();
}
public void ondevice()
{
PresentParameters pp = new PresentParameters
();
pp.SwapEffect = SwapEffect.Discard;
pp.Windowed = true;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat = DepthFormat.D16
;
device = new Device (0,DeviceType.Hardware
,this,CreateFlags.SoftwareVertexProcessing ,pp);

}
public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,-20),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.Transform.World = Matrix.RotationAxis
(new Vector3 (0,3,0),angle) * Matrix.Scaling (1,1,1)* Matrix.Translation
(posx,posy,posz) ;

device.RenderState.CullMode = Cull.None;
device.RenderState.Lighting =true;
device.RenderState.Ambient = Color.White     ;
}
public void meshh()
{
ExtendedMaterial[] material;

mesh = Mesh.FromFile (Application.StartupPath
+ @"\..\..\CAR.x",MeshFlags.SystemMemory ,device,out material);
mat = new Material [material.Length];
for (int i= 0;i < material.Length ;i++)
{
mat[i] = material[i].Material3D   ;
mat[i].Ambient = mat[i].Diffuse ;
}
}
```

```
public void light()
{
Color col = Color.White ;
Material mm = new Material ();
mm.Ambient = col;
mm.Diffuse = col;
device.Material = mm;

device.Lights [0].Diffuse = Color.White;
device.Lights [0].Type = LightType.Directional ;
device.Lights [0].Direction = new Vector3 (0,0,2);
device.Lights [0].Commit ();
device.Lights [0].Enabled = true;
}
public void render()
{
device.Clear (ClearFlags.Target |
ClearFlags.ZBuffer ,Color.Blue ,1,1);

device.BeginScene ();

for (int i=0;i < mat.Length ;i++)
{
device.Material = mat[i];
mesh.DrawSubset (i);
}
device.EndScene ();
device.Present ();
angle += 0.05f;
}
protected override void OnPaint
(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}


protected override void OnKeyDown
(System.Windows.Forms.KeyEventArgs e)
{
switch (e.KeyCode )
{
case Keys.Right :
posx ++;
break;

case Keys.**** :
posx --;
break;
```

```
case Keys.Up :
	posy ++;
	break;

case Keys.Down :
	posy --;
	break;

case Keys.Home :
	posz ++;
	break;

case Keys.End :
	posz --;
	break;

}


}

protected override void Dispose( bool disposing )
{
	if( disposing )
	{
		if (components != null)
		{
			components.Dispose();
		}
	}
	base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
	this.components = new
System.ComponentModel.Container();
	this.Size = new System.Drawing.Size(300,300);
	this.Text = "Form1";
}
#endregion

static void Main()
{
```

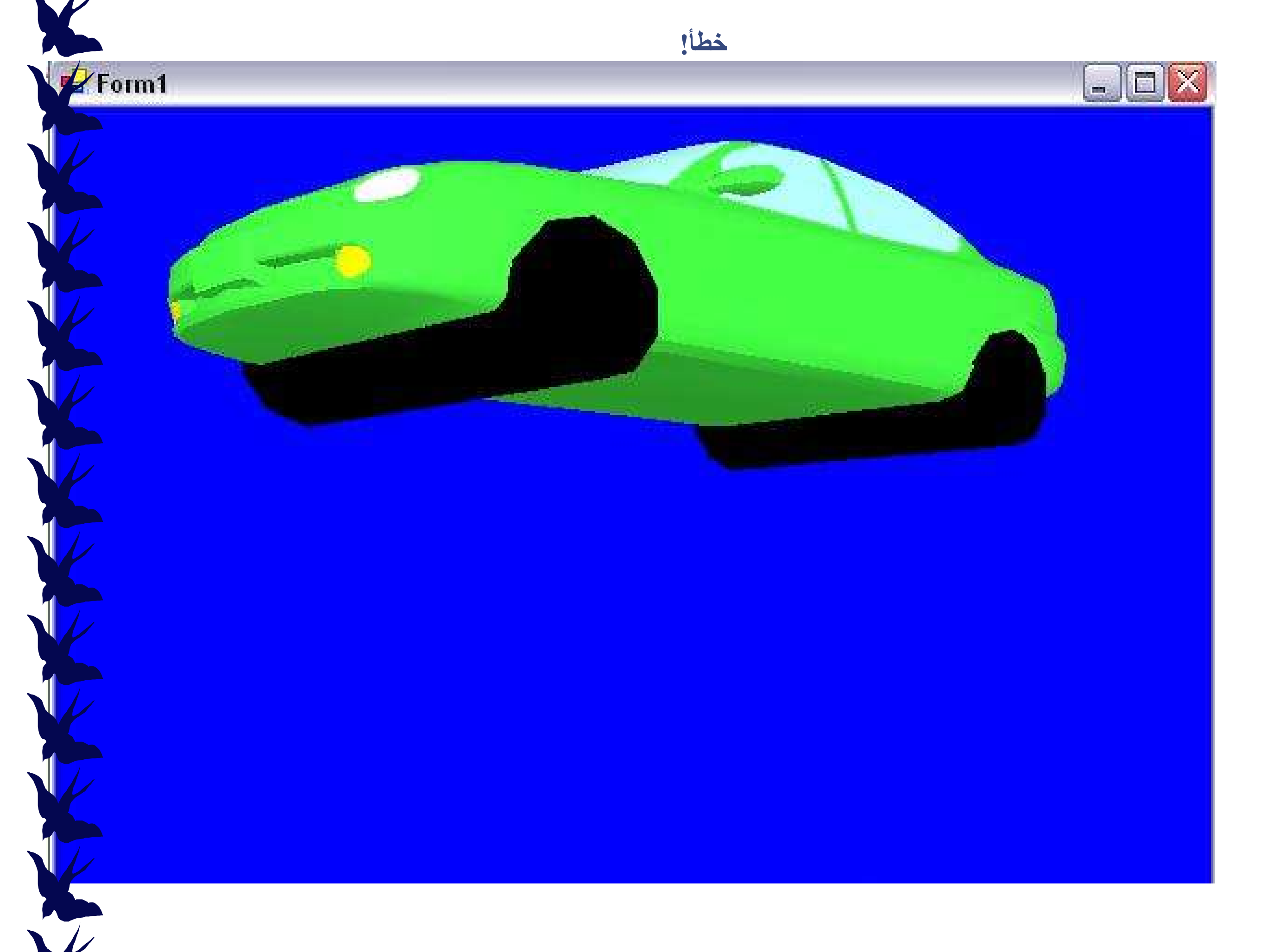
```
using (Form1 xx = new Form1 ())
{
    xx.ondevice ();
    xx.Show ();

    while (xx.Created )
    {
        xx.camera ();
        xx.meshh ();
        xx.light ();
        xx.render ();
        Application.DoEvents ();
    }
}

}
}
}
```

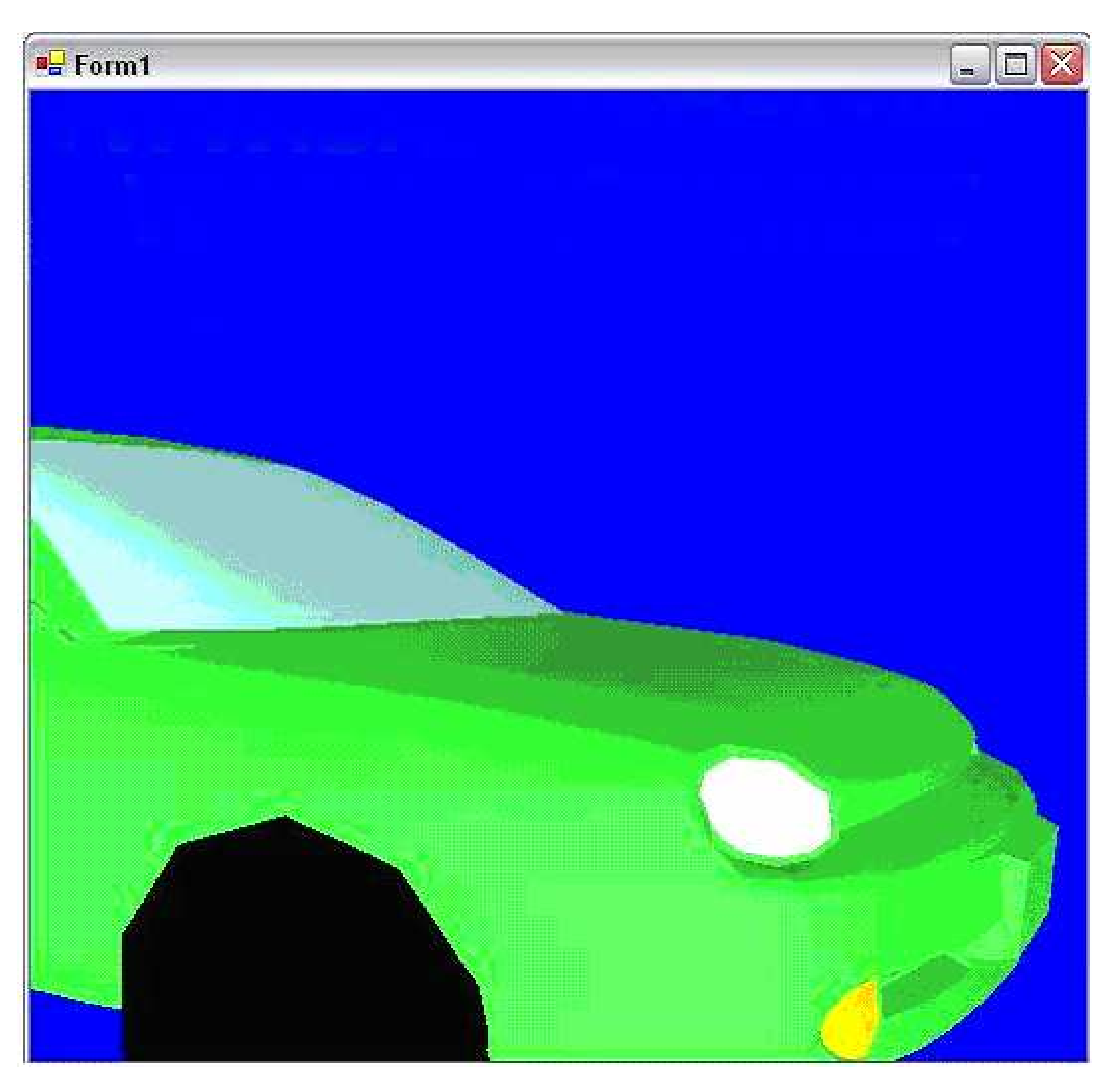
عند عمل RUN سيظهر الشكل بالأسفل :

خطأ!



لتحريك هذا الشكل:
يمين إضغط على زر السهم الأيمن من لوحة المفاتيح.

يسار إضغط على زر السهم الأيسر من لوحة المفاتيح.
فوق إضغط على زر السهم الأعلى من لوحة المفاتيح.
تحت إضغط على زر السهم الأسفل من لوحة المفاتيح.
لتكبير الجسم إضغط على زر Home من لوحة المفاتيح.
لتصغير الجسم إضغط على زر End من لوحة المفاتيح.



ملاحظة:
كل ما تكلمنا عنه في الدروس السابقة هو عن الـ Direct٣D وهو العنصر في الـ DirectX الذي يتيح لنا التحكم في كرت الشاشة (قلنا هذا سابقاً).. وهناك أيضاً عناصر أخرى سنتكلم عنها في الدروس القادمة منها الـ DirectSound وهو الذي يتيح لي التحكم في كرت الصوت... ما الذي أريد قوله .... بأن الـ Direct٣Dوالـ DirectSound يوجد بهما الكثير من الـ Class والـ Function المتشابهه في الأسم .... مما يسبب إرباك للـ DirectX على سبيل المثال هناك Class إسمها Device في الـ Direct٣D وأيضاً في الـ DirectSound, فأيهما يقصد المبرمج هل كرت الشاشة أم كرت الصوت ...
لحل هذه المشكلة, نقوم بعمل Namespace خاصة لكل عنصر أي:

كود:

```
using dev =  Microsoft.DirectX.Direct3D;
using snd = Microsoft.DirectX.DirectSound;
```

الآن أصبح الـ dev.Device يمثل الـ Device الخاص بي الـ (Direct٣D أي كرت الشاشة .)
وأصبح الـ snd.Device يمثل الـ Device الخاص بي الـ (DirectSound أي كرت الصوت.)
وعليه فإن الكود الذي كتبناه بالأعلا (السيارة) سيصبح:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using Microsoft.DirectX;
using dev =  Microsoft.DirectX.Direct3D;

namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private dev.Device  device;
private float angle;
private dev.Mesh  mesh;
private dev.Material [] mat;
private System.ComponentModel.Container
components = null;
float posx = 0.7f;
float posy = 0.7f;
float posz = 0.7f;



public Form1()
{
InitializeComponent();
}

public void ondevice()
{
dev.PresentParameters pp = new
dev.PresentParameters ();
pp.SwapEffect = dev.SwapEffect.Discard;
```

```
pp.Windowed = true;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat =
dev.DepthFormat.D16 ;
device = new dev.Device
(0,dev.DeviceType.Hardware
,this,dev.CreateFlags.SoftwareVertexProcessing ,pp);

}


public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,-20),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.Transform.World = Matrix.RotationAxis
(new Vector3 (0,3,0),angle) * Matrix.Scaling (1,1,1)* Matrix.Translation
(posx,posy,posz) ;

device.RenderState.CullMode = dev.Cull.None;
device.RenderState.Lighting =true;
device.RenderState.Ambient = Color.White     ;
}
public void meshh()
{
dev.ExtendedMaterial[] material;

mesh = dev.Mesh.FromFile
(Application.StartupPath +
@"\..\..\CAR.x",dev.MeshFlags.SystemMemory ,device,out material);
mat = new dev.Material [material.Length];
for (int i= 0;i < material.Length ;i++)
{
mat[i] = material[i].Material3D   ;
mat[i].Ambient = mat[i].Diffuse ;
}
}


public void light()
{
Color col = Color.White ;
dev.Material mm = new dev.Material ();
mm.Ambient = col;
mm.Diffuse = col;
device.Material = mm;

device.Lights [0].Diffuse = Color.White;
```

```
device.Lights [0].Type =
dev.LightType.Directional ;
device.Lights [0].Direction = new Vector3 (0,0,2);
device.Lights [0].Commit ();
device.Lights [0].Enabled = true;
}
public void render()
{
device.Clear (dev.ClearFlags.Target |
dev.ClearFlags.ZBuffer ,Color.Blue ,1,1);

device.BeginScene ();

for (int i=0;i < mat.Length ;i++)
{
device.Material = mat[i];
mesh.DrawSubset (i);
}
device.EndScene ();
device.Present ();
angle += 0.05f;
}
protected override void OnPaint
(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}


protected override void OnKeyDown
(System.Windows.Forms.KeyEventArgs e)
{
switch (e.KeyCode )
{
case Keys.Right :
posx ++;
break;

case Keys.**** :
posx --;
break;

case Keys.Up :
posy ++;
break;

case Keys.Down :
posy --;
break;
```

```
case Keys.Home :
        posz ++;
        break;

case Keys.End :
        posz --;
        break;

    }


}

protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.components = new
System.ComponentModel.Container();
    this.Size = new System.Drawing.Size(300,300);
    this.Text = "Form1";
}
#endregion

static void Main()
{
    using (Form1 xx = new Form1 ())
    {
        xx.ondevice ();
        xx.Show ();

        while (xx.Created )
        {
            xx.camera ();
```

```
xx.meshh ();
xx.light ();
xx.render ();
Application.DoEvents ();
}
}

}
}
}
```

-الـFont

أقصد الكتابة (Text) في الـ Direct٣d , فيما سبق كانت عملية الكتابة لا تتم مباشرة بل كانت بواسطة عمل texture أو Mesh. أما الآن فقد وفر علينا الـ net.هذا العناء بواسطة إستخدام هذه الفئة System.Drawing.Font كالتالي:
أولاً: نقوم بالتصريح عن متغيران:

كود:

```
private Font winfont;
private dev.Font  dxfont;
```

الأول (winfont): لإستخدام الحروف الموجودة في الـwindows
الثاني (dxfont): أخذ الحروف من الـ (winfont) وإستخدامها للكتابة في الـ DirectX

ثانياً: تعريف الـ(winfont نوع الخط, الحجم, والستايل) كالتالي:

كود:

```
winfont = new System.Drawing.Font
("Arial",20,System.Drawing.FontStyle.Bold);
```

ثالثاً :تعريف الـ) dxfont وضع خصائص الـ winfont بداخلة).

كود:

```
dxfont = new dev.Font(device,winfont);
```

رابعاً :عمل مستطيل وهمي (شفاف) لتحديد مكان توضع الـ) text يبدأ من أعلى يسار الشاشة) كما في الـ) Pixel تكلمنا عنه في الدروس السابقة).

كود:

```
Rectangle rr = new Rectangle (0,0,0,0);
```

تحديد النص المراد كتابته بداخل المستطيل الوهمي.

كود:

```
dxfont.DrawText ("WWW.3ASFH.NET",rr,dev.DrawTextFormat.NoClip
,Color.DarkOrange  );
```

تعمل هذه الخاصية DrawTextFormat على تحديد كيفية توضع الحروف بداخل المستطيل الوهمي) .أي هل يكبر المستطيل كلما كبرت الكلمة أم لا, هل يأخذ كامل الكلمة, هل توضع الكلمة يكون إلى اليسار أم اليمين أم في الوسط) تستطيع تجربتها لترى الفارق, أنظر إلى الجدول بالأسفل لتوضيح الفارق بينها:

Value	Purpose
WordBreak	If the text is longer than the box you give it, this option will break the text up onto the next line or however many lines it needs to fit the entire text.
VerticalCenter	Centers the text vertically
Top	Forces the text to be drawn at the top of the rectangle (default)
Bottom	Forces the text to be drawn at the bottom of the rectangle
SingleLine	Forces text to be drawn on a single line, ignoring newline characters.
NoClip	By default, the Font class clips the text into the rectangle you provide, so nothing is drawn outside the box. This disables that feature and makes drawing faster.
Center	Centers the text horizontally
Right	Forces the text to be drawn at the right margin
Left	Forces the text to be drawn at the left margin (default)

الكود كاملاً:

كود:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using Microsoft.DirectX;
using dev =  Microsoft.DirectX.Direct3D;

namespace WindowsApplication6
{

public class Form1 : System.Windows.Forms.Form
{
private dev.Device  device;
private float angle;
private dev.Mesh  mesh;
private dev.Material [] mat;
```

```
private System.ComponentModel.Container
components = null;
float posx = 0.7f;
float posy = 0.7f;
float posz = 0.7f;

private Font winfont;
private dev.Font  dxfont;


public Form1()
{
InitializeComponent();
}

public void ondevice()
{
dev.PresentParameters pp = new
dev.PresentParameters ();
pp.SwapEffect = dev.SwapEffect.Discard;
pp.Windowed = true;
pp.EnableAutoDepthStencil = true;
pp.AutoDepthStencilFormat =
dev.DepthFormat.D16 ;
device = new dev.Device
(0,dev.DeviceType.Hardware
,this,dev.CreateFlags.SoftwareVertexProcessing ,pp);

}

public void font()
{
winfont = new System.Drawing.Font
("Arial",20,System.Drawing.FontStyle.Bold);
dxfont = new dev.Font(device,winfont);
Rectangle rr = new Rectangle (0,0,0,0);
Rectangle rr2 = new Rectangle(0,30,0,0);
dxfont.DrawText
("WWW.3ASFH.NET",rr,dev.DrawTextFormat.NoClip ,Color.DarkOrange
);
dxfont.DrawText
("RAAD",rr2,dev.DrawTextFormat.NoClip ,Color.White  );

}

public void camera()
{
device.Transform.Projection =
Matrix.PerspectiveFovLH ((float)Math.PI /4,this.Width /this.Height ,1,50);
```

```
device.Transform.View = Matrix.LookAtLH (new
Vector3 (0,0,-20),new Vector3 (0,0,0),new Vector3 (0,1,0));
device.Transform.World = Matrix.RotationAxis
(new Vector3 (0,3,0),angle) * Matrix.Scaling (1,1,1)* Matrix.Translation
(posx,posy,posz) ;

device.RenderState.CullMode = dev.Cull.None;
device.RenderState.Lighting =true;
device.RenderState.Ambient = Color.White     ;
}
public void meshh()
{
dev.ExtendedMaterial[] material;

mesh = dev.Mesh.FromFile
(Application.StartupPath +
@"\..\..\CAR.x",dev.MeshFlags.SystemMemory ,device,out material);
mat = new dev.Material [material.Length];
for (int i= 0;i < material.Length ;i++)
{
mat[i] = material[i].Material3D   ;
mat[i].Ambient = mat[i].Diffuse ;
}
}


public void light()
{
Color col = Color.White ;
dev.Material mm = new dev.Material ();
mm.Ambient = col;
mm.Diffuse = col;
device.Material = mm;

device.Lights [0].Diffuse = Color.White;
device.Lights [0].Type =
dev.LightType.Directional ;
device.Lights [0].Direction = new Vector3 (0,0,2);
device.Lights [0].Commit ();
device.Lights [0].Enabled = true;
}
public void render()
{
device.Clear (dev.ClearFlags.Target |
dev.ClearFlags.ZBuffer ,Color.DarkSlateBlue ,1,1);

device.BeginScene ();
this.font ();
for (int i=0;i < mat.Length ;i++)
{
```

```
device.Material = mat[i];
mesh.DrawSubset (i);
}
device.EndScene ();
device.Present ();
angle += 0.05f;
}
protected override void OnPaint
(System.Windows.Forms.PaintEventArgs e)
{
this.render ();
}


protected override void OnKeyDown
(System.Windows.Forms.KeyEventArgs e)
{
switch (e.KeyCode )
{
case Keys.Right :
posx ++;
break;

case Keys.**** :
posx --;
break;

case Keys.Up :
posy ++;
break;

case Keys.Down :
posy --;
break;

case Keys.Home :
posz ++;
break;

case Keys.End :
posz --;
break;

}


}

protected override void Dispose( bool disposing )
{
```

```
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not
modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
this.components = new
System.ComponentModel.Container();
this.Size = new System.Drawing.Size(300,300);
this.Text = "Form1";
}
#endregion

static void Main()
{
using (Form1 xx = new Form1 ())
{
xx.ondevice ();
xx.Show ();

while (xx.Created )
{
xx.camera ();
xx.meshh ();
xx.light ();
xx.render ();
Application.DoEvents ();
}
}

}
}
}
```

عند عمل RUN سيظهر الشكل التالي:

Form1

